UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TECNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

TITULO: Control Remoto de Servicios para Espacios Interactivos Basado en NFC

NOMBRE: Iván Sánchez Milara

AÑO: 2013

Alumno: Iván Sánchez Milara.
Título del PFC: Control Remoto de Servicios para Espacios Interactivos basado en NFC.
Tutor: Jukka Riekki
Institución de acogida: Universidad de Oulu, Oulu, Finlandia.
Fecha de defensa: 18 Noviembre 2013

RESUMEN

La computación ubicua trae consigo una nueva era en la historia de la informática y las comunicaciones. Este nuevo paradigma surge como una evolución del ordenador personal (PC) en el que los usuarios ya no interaccionan con aplicaciones a través de un único ordenador. La capacidad computacional está distribuida en múltiples dispositivos y la inteligencia de los sistemas está repartida entre diferentes objetos que pueblan el entorno del usuario. Por tanto, la computación ubicua permite el control de servicios informáticos a través de la interacción con objetos próximos al usuario. La interacción hombre-máquina se hace así más natural aproximándose más a la forma en la que los humanos se comunican entre sí y con su entorno. Esto trae consigo la necesidad de redefinir la manera en la que el usuario solicita información a un sistema informático. El clásico paradigma de interacción entre el ser humano y ordenadores denominado WIMP (por sus siglas en ingles de Ventana, Icono, Menú y Puntero) ya no es válido. Los llamados entornos inteligentes abordan este problema tratando de predecir las intenciones y necesidades del usuario basándose en los datos capturados por sensores que se encuentran en el mismo espacio que el usuario. El principal problema con el que se enfrentan este tipo de sistemas es la complejidad de la comunicación humana. Es imposible, con el nivel tecnológico actual, modelar completamente la manera en que los seres humanos interactúan con su entorno. En consecuencia, el sistema podría interpretar erróneamente la información recibida de los sensores y actuar de una manera no esperada por el usuario, llegando incluso a efectuar acciones contrarias a sus deseos. El usuario pierde así la sensación de control sobre el sistema lo que lo podría generar un sentimiento de frustración que conllevaría a una percepción muy negativa del mismo. Por tanto, es necesario involucrar más al usuario en el proceso de toma de decisiones así como proveerle de las herramientas necesarias para que pueda corregir decisiones erróneas hechas por el sistema. En este contexto podemos definir una alternativa a los entornos inteligentes: los espacios interactivos. En ellos el sistema, en vez de predecir las intenciones del usuario basándose en los datos recogidos por los sensores, responde a acciones explicitas (y previamente definidas) realizadas por el usuario.

Los espacios interactivos se sustentan en dos pilares fundamentales: (1) los usuarios tienen siempre el control del sistema, y éste no inicia ninguna tarea sin que el usuario de la correspondiente orden a través de una acción explicita (y que previamente ha sido almacenada en el mismo) y (2) el usuario interactúa con los servicios ofrecidos en un entorno a través de los objetos que se encuentran en dicho entorno, y no a través de un ordenador. En cierta forma, los espacios interactivos tratan de trasladar el estilo de interacción WIMP al entorno del usuario, alojando los distintos componentes del interfaz en distintos elementos del entorno del usuario.

En este PFC presento REACHeS (Remote Enabling And Controlling Heterogenous Services traducido al español como Activación y Control Remoto de Servicios Heterogéneos). REACHeS es una plataforma que posibilita la creación de espacios interactivos habilitando la comunicación entre servicios, dispositivos (como por ejemplo pantallas y altavoces) y usuarios que se encuentran en un mismo entorno. REACHeS permite a los usuarios controlar aplicaciones que corren en un servidor remoto a través de clientes de software que generalmente están instalados en teléfonos móviles. Estas aplicaciones, a su vez, pueden controlar los dispositivos necesarios para proveer al usuario del correspondiente servicio. Los clientes, servicios y dispositivos no se comunican directamente sino que REACHeS hace de pasarela entre ellos. Entre las principales funcionalidades de REACHeS destaca el registro de servicios y dispositivos, el control de la sesión del cliente, la asignación dinámica de dispositivos a servicios, el control, manejo y filtrado de errores así como la adaptación de la respuesta de los servicios a las características de los clientes. REACHeS usa el protocolo HTTP para transmitir mensajes entre los distintos elementos del sistema.

Los usuarios controlan los servicios y dispositivos registrados en REACHeS interactuando con objetos cotidianos de su entorno, lo que genera un interfaz de usuario distribuido y alojado en el propio entorno. Normalmente estos objetos no tienen capacidad de computación. Además, no se pueden comunicar ni con otros objetos ni con ningún sistema informático. La Comunicación de Campo Cercano (NFC por sus siglas en inglés) es una tecnología que permita aportar esas capacidades a cualquier elemento del entorno del usuario. NFC es una tecnología emergente compatible con algunas soluciones basadas en RFID. Las etiquetas NFC son unos dispositivos muy delgados que se pueden acoplar fácilmente a cualquier objeto y que permiten almacenar permanentemente cierta cantidad de datos. Estos datos pueden ser leídos por un lector de NFC cuando éste se aproxima suficientemente a la etiqueta. Actualmente, existe un importante número de teléfonos móviles que incorporan un lector de NFC, por lo que dicho teléfono puede leer el contenido de una etiqueta NFC.

REACHeS usa la tecnología NFC para construir interfaces alojados en el entorno del usuario. Para ello, se almacenan comandos y parámetros asociados a los mismos en etiquetas NFC que se adhieren a diferentes objetos del espacio de usuario. Los parámetros dependen del objeto en el que se encuentre localizada la etiqueta. Cuando el usuario toca la etiqueta con un teléfono móvil que integre la tecnología NFC, el comando se transmite a REACHeS que lo procesa y lo reenvía al correspondiente servicio. Un importante punto a tener en cuenta en el diseño de este tipo de interfaces es la manera de anunciar al usuario donde se encuentra una etiqueta NFC y qué comando se va a enviar a REACHeS cuando se toca dicha etiqueta. Nosotros proponemos poner iconos sobre la etiqueta NFC. Los iconos son fácilmente visibles por el usuario y su pictograma muestra el comando a ejecutar. En cierta medida, equivale a trasladar los iconos de los interfaces WIMP al entorno del usuario. La función de los iconos es la misma en ambos casos: ejecutar ciertos comandos.

Un escenario general para REACHeS sería el siguiente: un usuario solicita un determinado servicio tocando el correspondiente icono con un teléfono móvil que incorpora la tecnología NFC. El icono se encuentra en algún elemento del entorno del usuario. El teléfono móvil arranca el cliente asociado a dicho servicio y envía el comando "*inicia el servicio X*" a REACHeS dentro de una solicitud HTTP. REACHeS realiza la asignación de los dispositivos que el servicio necesita y reenvía el comando al servicio. El servicio procesa el comando y genera una respuesta HTTP. Simultáneamente el servicio puede enviar comandos a los dispositivos que le han

asignados. El cliente procesa la respuesta y modifica su interfaz de usuario convenientemente. El usuario puede enviar nuevos comandos al servicio tocando más iconos con su teléfono móvil. Los comandos se empaquetan en una petición HTTP y se envían a REACHeS, que se encarga de redirigirlos al servicio adecuado. La respuesta generada por el servicio se devuelve al cliente. Este proceso se repite hasta que el usuario envía el comando de "*detener el servicio X*". REACHeS interrumpe el servicio y libera los dispositivos asociados para que otros servicios puedan hacer uso de ellos.

La principal contribución de este PFC es el diseño e implementación de REACHeS, una plataforma que permite la creación de espacios interactivos y por tanto el control de servicios y dispositivos interactuando con objetos del entorno del usuario. Además este PFC estudia cómo se puede usar la tecnología NFC para construir interfaces de usuario para espacios interactivos. REACHeS usa NFC para iniciar y controlar servicios, así como para seleccionar e interactuar con dispositivos. El análisis de viabilidad y limitaciones de REACHeS, así como de distintos aspectos relacionados con la interacción de los usuarios dentro de espacios interactivos se estudia a partir de prototipos que se han implementado y testado. Dichos prototipos y tests de usuario se presentan también en este documento. Finalmente, en la parte teórica, se redefine el concepto de espacio interactivo y se estudian distintos modos de interacción y distintas tecnologías que se pueden usar para construir este tipo de entornos.

OULUN YLIOPISTO
UNIVERSITY of OULU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

IVÁN SÁNCHEZ MILARA

NFC BASED REMOTE CONTROL OF SERVICES FOR INTERACTIVE SPACES

Master's Thesis
November 2013

Sánchez Milara, Iván (2013) NFC Based Remote Control of Services for Interactive Spaces. Technical University of Madrid. Higher Technical School of Telecommunications Engineering. Master's thesis, 122 pages.

ABSTRACT

Ubiquitous computing (one person, many computers) is the third era in the history of computing. It follows the mainframe era (many people, one computer) and the PC era (one person, one computer). Ubiquitous computing empowers people to communicate with services by interacting with their surroundings. Most of these so called smart environments contain sensors sensing users' actions and try to predict the users' intentions and necessities based on sensor data. The main drawback of this approach is that the system might perform unexpected or unwanted actions, making the user feel out of control. In this master thesis we propose a different procedure based on Interactive Spaces: instead of predicting users' intentions based on sensor data, the system reacts to users' explicit predefined actions. To that end, we present REACHeS, a server platform which enables communication among services, resources and users located in the same environment. With REACHeS, a user controls services and resources by interacting with everyday life objects and using a mobile phone as a mediator between himself/herself, the system and the environment. REACHeS' interfaces with a user are built upon NFC (Near Field Communication) technology. NFC tags are attached to objects in the environment. A tag stores commands that are sent to services when a user touches the tag with his/her NFC enabled device. The prototypes and usability tests presented in this thesis show the great potential of NFC to build such user interfaces.

Key words: interaction, physical user interfaces, tangible user interfaces, ubiquitous computing, server platform.

Sánchez Milara, Iván (2013) NFC:hen pohjautuva palveluiden kauko-ohjaus interaktiivisille tiloille. Madridin teknillinen yliopisto. Tietoliikennetekniikan tiedekunta. Diplomityö, 122 s.

TIIVISTELMÄ

Jokapaikan tietotekniikka (yksi ihminen, monta tietokonetta) on kolmas aikakausi tietojenkäsittelyn historiassa. Se seuraa ensimmäistä, keskustietokoneiden aikakautta (paljon ihmisiä, yksi tietokone) ja toista, PC:n aikakautta (yksi ihminen, yksi tietokone). Jokapaikan tietotekniikan myötä digitaalisten palveluiden hallinta tapahtuu enenevässä määrin hallitsemalla käyttäjää ympäröivää ympäristöä. Suurin osa toteutetuista, niin kutsutuista älykkäistä ympäristöistä, sisältää sensoreita, jotka tarkkailevat käyttäjän toimintaa ja yrittävät ennustaa hänen aikomuksiaan ja tarpeitaan. Tämän lähestymistavan suurimpana varjopuolena on, että järjestelmä saattaa suorittaa yllättäviä ja ei toivottuja toimintoja ja saada käyttäjän näin tuntemaan, ettei hän hallitse toimintoja. Tässä diplomityössä esitellään erilainen toimintatapa interaktiivisille ympäristöille. Järjestelmä ei pyri ennustamaan sensoreista saadun tiedon perusteella käyttäjäjän aikomuksia, vaan reagoi käyttäjän interaktiivisessa ympäristössä tekemiin toimintoihin. Tässä työssä toteutetaan REACHeS-palvelinalusta, joka yhdistää samassa ympäristössä olevat palvelut, resurssit ja käyttäjiät. REACHeS:in avulla käyttäjä hallitsee palveluita ja resursseja olemalla vuorovaikutuksessa häntä ympäröivien jokapäiväisten esineiden kanssa ja käyttämällä matkapuhelinta käyttäjän, ympäristön ja järjestelmän välisen vuorovaikutuksen välittäjänä. REACHeS:in käyttöliittymät on rakennettu hyödyntäen NFC (Near Field Communication) -teknologiaa. NFC-tunniste voidaan kiinnittää mihin tahansa esineeseen ympäristössä. Tunnisteeseen on tallennettu komento, joka lähetetään palvelulle, kun käyttäjä koskettaa tunnistetta NFC-lukijalla varustetulla laitteella. Tässä diplomityössä esitetyt prototyypit ja käyttäjätestaukset osoittavat, että NFC soveltuu hyvin tällaisten käyttöliittymien rakentamiseen.

Avainsanat: vuorovaikutus, fyysiset käyttöliittymät, jokapaikan tietotekniikka, palvelinalusta.

TABLE OF CONTENTS

FOREWORD

This Master's Thesis summarizes my research work carried out for several years in the Department of Computer Science and Engineering department at the University of Oulu. This thesis serves to conclude my studies in Telecommunication Engineering in the Universidad Politécnica de Madrid.

First of all, I would like to express my sincere gratitude to my thesis supervisor Prof. Jukka Riekki for providing me guidance and support for my work. I also appreciate that you gave me the opportunity to start my research career in the University of Oulu. I would like to express my thanks also to Dr. Susanna Pirttikangas for reviewing and evaluating my work, as well as for your support in the initial steps of my research work.

Secondly, I would like to acknowledge all my colleagues from CSE who have contributed in the development of the different prototypes and concepts presented in this work. I would like to express my gratitude especially to Mikko Pyykkönen and Oleg Davidyuk whose work and support have been decisive for finishing this Master Thesis. I acknowledge also Hannu Rautio and Jukka Kontinen for their efficient technical support.

Thirdly, I would like to thank my parents who gave me the chance to complete my studies in Oulu.

Finally, I would like to thank especially to Marta my life partner and colleague, who has had a definitive role in this thesis. You have contributed through your own work but also through your support, encouragement and patience. Seriously, many thanks for everything. Of course, I cannot forget Niko who has been with us the last two years. Many thanks for … being there.

Oulu, 16th November 2013

Iván Sánchez Milara

LIST OF ABREVIATIONS AND SYMBOLS

NFC	Near Field Communication. Technology enabling the communication between two devices when they are brought into close proximity.
RFID	Radio Frequency Identification. Technology that uses radio frequency electromagnetic fields to transfer data from two devices with the purpose of identifying one of them.
REACHeS	Remote Enabling and Controlling Heterogeneous Services. Server platform that communicates services and resources with users and their environments.
AmI	Ambient Intelligence. Environments that sense and response to people's stimulus to support them in their daily activities.
SmE	Smart Environment. Environments composed by sensors, actuators and computational elements seamlessly embedded in everyday life objects.
IoT	Internet of Things. Broad term used to define the set of technologies that enables the communication among objects in the environments by assigning them a unique id.
UI	User interface. The interaction space between humans and machines.
GUI	Graphical User Interface. User interfaces based on graphical symbols and visual indicators.
WIMP	Windows, Icon, Menus and Pointer. Classical style of interactions, based on those elements, that is still the main interaction method in computers, tablets and phones.
PUI	Physical User Interface. Interaction style which uses physical objects to interact with machines.
TUI	Tangible User Interface. Synonym of PUI.
SUI	Surface User Interface. Interaction style in which users interact with the environment through big flat surfaces usually using their fingers.
NUI	Natural User Interface. Interaction style based on humans natural interaction (e.g. voice and gestures).
ISO	International Organization for Standardization.
IEC	International Electrotechnical Commision
XML	Extensible Markup Language. Text based serialization language with markup notation.
HTTP	Hypertext Transfer Protocol. The most used application layer protocol for hypermedia information systems.

1 INTRODUCTION

1.1 Background

Mark Weiser coined the term “Ubiquitous Computing” in his famous article “The Computer for the 21st Century“ [1]. Weiser envisioned a world in which computers vanish themselves in the environment, integrating seamlessly with the world: “*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*”. We can infer two ideas from Weiser’s vision of Ubiquitous Computing: (1) every object in the world can be seen as a computer, that is, to have computing intelligence, and (2) humans do not interact with those objects as if they were computers, but as they would do with normal objects. This is known as natural interaction [2]. The concept of calm technology allows a user to utilize the computer capabilities of the objects when performing her everyday life activities. The technology is hidden in a second plane of the user reality. This concept is similar to walking in the street and coming up with an advertisement in a poster. Unintentionally, you assimilate the information in the poster, although you do not read it consciously.

During the latest twenty years many researchers have tried to create systems that accomplish Weiser’s vision of calm technology using different approaches, even providing different names for the same concept, although with different shades of meaning. IBM started in the mid 90’s the research on what they call Pervasive Computing. MIT media lab and ETH Zurich have created their own divisions to study Wearable Computing. In this case, users carry themselves the devices which communicate with the surroundings. Others prefer to give more importance to the fact that the technology is hidden in the environment calling it Ambient Intelligence. Greenfield [3] prefers to use the term Everyware, since the technology is distributed everywhere.

One big challenge of Ubiquitous Computing is to make the communication between human and computers as close as possible to the communication between humans [4]. Humans have such an advanced communication and interaction system that it is impossible to model completely. In human interaction there are many factors to take into account: shared knowledge and common understating on how the world works, ability to detect errors and recover from them by monitoring the response from the communication partner, capacity to analyze and modify dynamically the communication based on situation and context, use of multiple and simultaneous communication channels (voice, body language, gestures, voice pitch ….). Computers cannot understand our language, although nowadays voice recognition is quite advanced, our language: is not only based on transmitting monotonous sounds but we use pitch, different accents, and body language to communicate with others.

Traditional desktop and mobile computers, use the WIMP (Windows, Icon, Menus and Pointer) paradigm to communicate with users. A keyboard and a mouse provide explicitly input to the computers, in a language that the computers understand. This is far away from the *calm technology* that Weiser envisioned. Nowadays there are two main approaches to address this input deficiency [5]. The first one tries to improve the language that humans use to interact with computers while the second one tries to improve the efficiency of machines on getting and analyzing context information. In this Master Thesis we state that in long term the second approach fits better the Weiser’s vision of Ubiquitous Computing, however current technology is not able to

get and analyze the vast amount of context information that humans can. So, we have followed the first approach when trying to create environments that match in the Ubiquitous Computing paradigm. The second approach has its place in either Ambient Intelligence (AmI) or in Smart Environments (SmE) [6], [7]. We realize the first approach by removing a great part of the environment intelligence and bringing UIs to the environment itself. We call to this type of environments Interactive Spaces.

AmI and SmE concepts, although related are not exactly the same. A smart environment is a place that has been enriched with sensors, actuators and other devices interconnected through a network and controlled by a back-end system. The back-end system could be either centralized, be part of a distributed system or more recently live "in the cloud". Ambient Intelligence uses the different elements of the smart environments to provide a collective intelligence. A user perceives the intelligence of the environment as a whole and not as smartness of the different elements. In traditional AmI environments, a smart space collects implicit information about (i) the state of the environment and (ii) the state of the user using automated means. That is, the SmE is continuously sensing the context of the user, including all the objects in the environment. Context is, as Dey and Abowd have defined in [8]: "*any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*". AmI environments use context to provide relevant information and services to the user, where relevancy depends on the tasks [8]. Hence, the environment reacts to changes in the context. The same set of actions performed by the user, in the same space, could lead to the execution of different task or starting a different set of services depending on the context.

The big challenge is how to filter, analyze and interpret all data produced by sensors. We, humans, are really good at analyzing our context but machines are not. Usually, the different contexts must somehow be predicted by the application designer (even when there is reasoning involved). The designer decides what the relevant information is and how to deal with it. Another difficulty in this kind of systems is to assign user intentions to situations. When a user performs an action, how can the system infer which are the user expectations for that action? The response that the system designer associates to a user action is not the one that user is expecting at that moment. Even when using complicated systems such as neural networks, that utilize knowledge from past actions, is impossible to assure with a high percentage of reliability that the response meets the user expectations. The low capacity of the user to participate in the computation process and the decision making tasks provokes a bad user experience. The environment takes over the control of the full process, and the user is seldom explicitly involved in the interaction. Furthermore, the excessive hiding of the technology gives the user low possibilities to respond and react when she detects errors.

Related research shows that the user is willing to be included in the decision making tasks. In AmI environments, humans should be taken as another component of the system architecture [9]. Human centered AmI takes humans (and their abilities) as an important part of the system design. For example, why do we need an actuator to open a door, when humans can do it? These limitations force us to look for an alternative. One possibility is to define Interactive Spaces as the computer enhanced environments in which explicit interaction and user control are emphasized in comparison to the implicit interaction and context recognition emphasized in SmEs.

In an Interactive Space a user starts the interaction with the system explicitly, that is, performs a specific and predefined action in the environment. For example, to start an application the user manipulates (grasp, touch, squeeze, etc.) an object. The environment does not need to sense the user space continuously. When the user initiates the interaction, the system pulls the context and processes it. This is different from classical AmI environments where the context is pushed by the environment itself, using continuous sensing. Interactive Spaces do not try to hide the computation from the users, but just integrate it into the environment. The user is always aware that she is interacting with a computer application. The interaction cannot start without that user awareness. Furthermore, during the whole interaction the user is in control. Interactive Spaces emphasize user control; instead of studying solutions for deducing the users' goals automatically, the focus is on user interfaces that let the users inform their goals to the system. The system should provide feedback, and the user should be able of correct possible errors that the system makes during the execution of the service or the application.

The traditional interaction based on the WIMP paradigm is built upon an output device (usually a screen) and one or several input devices (keyboards, mice, trackballs, etc.). User can, for example, enter some text using the keyboard, access some specific function using combinations of keys (shortcuts), use the pointing device to browse menus and click on icons to start. Currently, in smartphones and tablets the concept is similar but the input device is a user's finger instead of a mouse. Users open applications by tapping icons on the screen and moves between different open applications by swiping the finger on the screen. Although the pointing device has changed, the UIs still follow the WIMP design principles. Traditionally, the HCI style that the WIMP paradigm embraces, requires that the user is in front of the computer to interact with the applications using a mouse (or any other pointer device) and a keyboard. It is far away from Weiser's idea of ubiquitous computing. In recent years, portable devices such as laptop computers, tablets and smart phones have changed this concept. Those devices permit interaction with services everywhere, hence bringing interaction a big step towards ubiquitous computing. However, the applications running on those devices still follow the WIMP paradigm. The UI is still on a device that the user has to carry. This is not what Interactive Spaces are aiming for.

In Interactive Spaces computation and interfaces for a user are embedded in the environment itself and not in some device that the user is carrying. Interaction with applications is not centralized into a single device, but every object in the environment can potentially give an input to applications or provide some output to users. It means using the real world as an interface to interact with services. Tangible bits "*allows users to grasp and manipulate bits in the center of users' attention by coupling the bits with everyday physical objects and architectural surfaces*" [10]. This concept led later to the proposition of Tangible User Interfaces (TUIs) that some authors also call Physical User Interfaces (PUIs). In those UIs, users utilize the affordance of an object to interact with services. Affordance is defined by Norman as "*perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could be possibly used"* [11]. The action performed in the physical world has potentially an effect in the digital world, for example when a student turns the page of her course book in a new lesson, a display might show a diagram with the main contents of the new chapter. Thus, computation is embedded and embodied in physical devices and appliances [12]. People have acquired dexterity and skills to manipulate objects in normal life. Interactive Spaces use that dexterity and skills to manipulate

computation devices, or even in more abstract terms services offered by the environment. With TUIs, we can go beyond the simulated manipulation of WIMP, where icons and windows are metaphors of objects in the real world. TUIs allow the user to directly manipulate a real object, which has two interfaces: one physical with the user and one digital with the system.

Using TUIs to completely remove GUIs presents two problems; firstly, objects must communicate with a computer system. Technically, it is not easy to add communication capabilities to multiple objects or appliances in the user environment when they are not designed for that. Secondly, the WIMP paradigm has a strong effect on how people interpret user interfaces. People assume that to interact with a computer application one should use a display where the system is providing feedback and a set of buttons and a pointing mechanism to provide input to the system. A user might be surprised when grasping an object in the environment produces an effect on a close by display, although the metaphor is clear, and would be natural if the users did not carry the WIMP prejudices. Until users overcome those prejudices, we should teach the user how to interact with these new environments. One option is to bring classical GUIs artifacts (icons, pointers) to the physical world. Icons embedded in the environment can be used to command services and devices. For that, we need a technology which links the icons with the system.

The Internet of Things (IoT) [13] shares similar communication problems as the ones presented for Interactive Spaces. The IoT is a "*a world-wide network of uniquely addressable and interconnected objects, based on standard communication protocols*" [14]. IoT uses Radio-Frequency identification (RFID) as enabler technology. RFID permits to connect every object in the user environment to a database or networks inexpensively, assigning to each object a unique id. RFID chips have unique ids and store a limited amount of data. A RFID reader can read both the data stored in the chip and the id wirelessly. RFID chips are easy to attach to objects and hence permit the identification of every object in the environment and make them discoverable by RFID readers. Finally, current advances in hardware development make possible the miniaturization of memory and processors.

NFC (Near Field Communication) is an emerging technology compatible with HF RFID (13.56 MHz) tags and readers. In comparison with HF RFID, NFC offers three different use modes: read/write mode, P2P mode and card emulation mode. The read/write mode is similar to reader/writer mode in classical RFID solutions. The P2P mode permits bidirectional communication between two devices. The card emulator mode permits using the mobile phone as a contactless smart card that can be used in payment and ticketing applications.

Embedding NFC technology into mobile phones opens a high range of possibilities when implementing new interfaces for Interactive Spaces. NFC tags can be embedded to objects in the environment while phones can be used as "*mediator*" between users and objects in the environment. Users interact with object by touching the NFC tags with an NFC enabled mobile phone. Tagging objects in the environment with NFC tags permits the creation of an innovative type of user interfaces that can be categorized as a TUI. NFC allows connecting objects and other devices in our everyday life environment to applications and systems in an inexpensive manner. NFC tags are slim and small devices that can be attached to almost any object. NFC chips can be embedded to various devices to create or improve their communication and interaction capabilities. Modern phones have enough processing power to perform complex operations. They also support network connectivity via Wi-Fi, 3G, 4G or

GPRS. The network capabilities permit not only running applications in the mobile phone, but also connecting to Web services running in external infrastructure. In the last case the phone has a client to communicate with Web services. User interacts with the applications using the UI provided by the phone. Additionally, in NFC enabled mobile phones, users touch objects in the environment, or other NFC enabled devices, to interact with the applications. The data stored on NFC tags identify the tasks the phone must execute. The phone is used as a mediator between the service, the user and the environment. However, the user experiences that the computing power is in the object itself and not in the mobile phone since applications react to user interaction. This opens an innovative way of communication between users, environment and applications that has not been widely explored yet, shifting the metaphors created for WIMP UIs to the user environment. Icons are moved from a screen to objects in the environment.

The main advantages of using NFC as a key technology for Interactive Spaces are:

1. NFC is embedded in the phone so there is no need to additional hardware. Mobile phones offer processing power, network communication and multimedia capabilities. Moreover, they have become an essential device for our everyday life. Furthermore, integration with the software is total: the application developer can use the NFC reader natively, accessing directly to the API provided by the manufacturer without need to install any external driver or application in the phone.
2. NFC Forum[1] has standardized both the format in which the data is stored in the tag and transmitted between NFC enabled pairs and the APIs to read/write NFC data. It facilitates the adoption of NFC technology by multiple platforms simplifying the compatibility among platforms.

We claim that NFC is an enabler technology to create for Interactive Spaces, a new generation of UIs which permit more natural interaction between users and applications.

1.2 Motivation and goal of the Thesis

Research is needed to confirm that Interactive Spaces, as described in the previous section, are suitable to build more user-centered pervasive computing environments. It is also necessary to study the user's and developer's acceptance of the Interactive Space's concept. It is also important to study which are the most suitable technologies and interaction methods for that kind of environments. Moreover, NFC has a high potential to be used as interaction technology for Interactive Spaces. However, there has not been too much research on how this technology should be deployed, which are the best ways of advertising tags to the users, how users accept the technology, or how NFC could be combined with other interaction modes such as classical GUIs, speech recognition, haptic feedback, gesture recognition, etc.

The Living Lab methodology is a good approach to study the previous research questions. MIT Living Labs[2] webpage states that "*Living Labs is a user-centric research methodology for sensing, prototyping, validating and refining complex solutions in multiple and evolving real life contexts.*" One possible implementation of this approach for this concrete problem is to first identify small scenarios where the new UIs can be used, then design and implement applications that fit those scenarios. After that, the users test the scenario and based on the results the application is

[1] http://www.nfc-forum.org/home/
[2] http://livinglabs.mit.edu/

modified. Finally, and after several iterations it is possible to extract design recommendations and probably build a framework to help application designers to deploy this new type of user interfaces in user environments as well as integrate it with existing applications. During the whole process, we emphasize the necessity of testing the different design concepts with fully working applications deployed in environments as close as possible to real ones.

Creating simple applications, fast to implement and deploy, is not easy for Interactive Spaces. This is because applications have multiple components distributed in different locations: clients running in mobile devices, services running on Internet and local resources located in the environment (displays, speakers, etc.) Furthermore services might need multiple resources simultaneously, and some of those might be locked by other users or services sharing the same environment. From these requirements it is easy to infer that there is a need of a server platform that connects services, resources and users with each other permitting communication among them as well as providing a resource allocation system which controls the owners of each resource. Thus, the goal of this Master Thesis is to design and implement such platform that I have named REACHeS (Remote Enabling and Controlling HEterogenous Services).

On the other hand, although NFC is an enabler technology for Interactive Spaces, due to its novelty, it has not been studied in detail from user interaction perspective. REACHeS platform permits the realization of user tests to analyze users' acceptance. The tests permit also gaining better knowledge on how NFC can be integrated as interaction technology in Interactive Spaces.

This Mater Thesis produces two main results: (1) the design and implementation of REACHeS platform and (2) the implementation and user testing of several NFC enabled applications for Interactive Spaces using REACHeS.

1.3 Structure of the Thesis

The rest of this Master Thesis is structured as follows: the second chapter presents in detail RFID and NFC technologies, their technical foundations as well as the main applications areas where they are used. In the third chapter I describe the main interaction technologies for Interactive Spaces. Later, in chapter four, I present the main contribution of this Master Thesis, the REACHeS platform. This chapter provides the technical details of the platform: architecture, functionality and implementation. In the next chapter, I show how REACHeS is integrated in Interactive Spaces. REACHeS is the link among the environment, the users and the services offered by that environment. I also present example applications that serve as proof of concepts. The usability studies are explained in the chapter six. Finally, chapter seven leads to discussion and comparison with related work. The Master Thesis concludes presenting the main achievements of the thesis and a short discussion of future work.

2 RFID AND NFC TECHNOLOGY

2.1 RFID and Smart Cards

NFC technology is an evolution of RFID technology working in HF permitting the integration of RFID in modern devices and systems. This section gives an overview of RFID technology. For more detailed information reader can check Finkenzeller's book [15].

2.1.1 Introduction

RFID technology uses radio waves transmission to identify, track or detect people and a wide variety of objects. This technology has been used in a wide range of applications areas such as security and access control, transportation, supply chain tracking, animal identification and tracking or ticketing. In recent years, this technology has gained importance due to the fact that is a key technology for the Internet of Things allowing the connection of any object to Internet. An RFID system (Figure 1) is always composed by at least two components: transponders or tags and readers (also known as interrogators or scanners). The readers send and receive RF data to and from the tag via coupling elements (antennas).

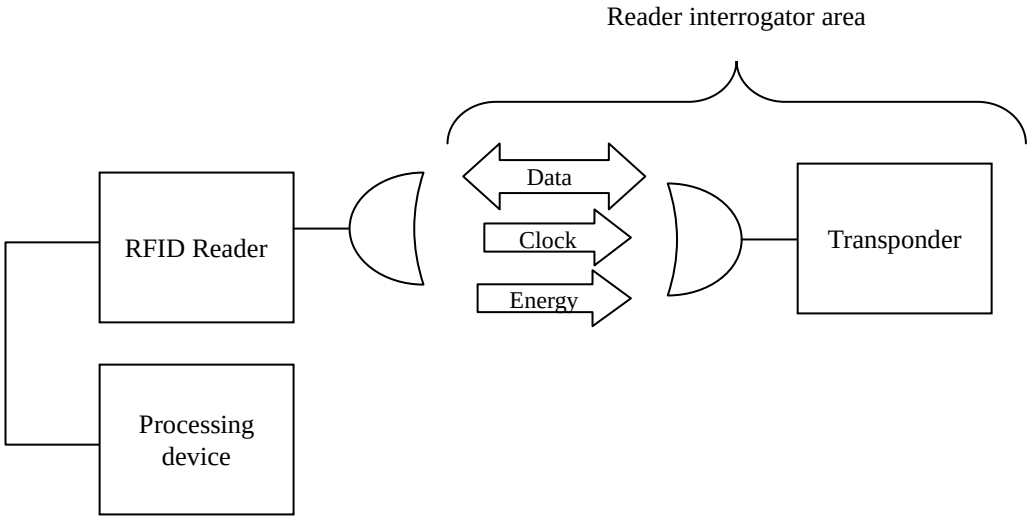


Figure 1. RFID system. Source: [15, p. 7].

A *transponder* is a device placed on the object or person to be identified or tracked. It is normally composed by a silicon microchip and a coupling element. There exists a particular case of transponders that do not contain a microchip. Some of them are named 1 bit transponder and they are mainly used for anti-theft devices at shops (EAS, electronic article surveillance system). This kind of transponder just announces its presence to a reader by using the properties of certain circuits and materials to alter one or several properties (magnitude, frequency, phase...) of the electrical or magnetic field generated at the reader.

The microchip contains a memory and a microprocessor. The memory stores the transponder data while the micro-processor controls the memory access (including authorization for writing and reading in memory and encryption). The transponders store in memory a unique serial number which identifies them uniquely (and so

identifies the object to which the transponder is attached to). Depending on the transponder technology and the type of memory used, the transponder can contain extra information associated to the object which the tag is attached to. The coupling element enables the RF-link to receive and transmit data from and to the reader.

Transponder comes in a wide variety of construction formats, permitting its use in a wide range of applications and facilitating the tagging of objects, devices, animals and people. Most common housing for transponders is ABS injection moldings with the shape of a coin, glass or plastic. The transponder can be also embedded in other objects such as keys, watches or credit cards. Smart labels, also known as RFID tags, are widely used nowadays. These sticky labels are very slim electronic components, in which the coupling element is a coil applied to a very thin plastic foil. The foil is laminated with paper and coated with adhesive (which permits stick them to any good or object). Figure 2 shows a Mifare 1K RFID tag.

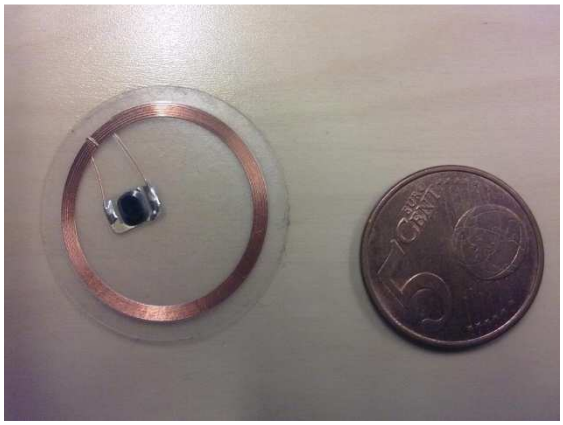


Figure 2. RFID tag.

RFID transponders may be classified following a wide set of criteria [16]. One very important feature of a transponder is if it has its own power supply or it needs to draw energy from an external source. On one hand, *active transponders* have their own transmitter and power source (e.g. battery or solar cell) for the operation of its microchip. They generate their own radio signal to transmit data from its microchip to the reader. The internal power supply permits longer actuation distances (around 100 m) and the usage of faster memories. Main drawbacks are its high manufacturing price (in comparison with passive transponder) and other problems associated to battery maintenance. Broadly speaking there are two kind of active transponders: those who are constantly sending radio beacons signals[3] and those who are activated only after receiving a signal from the reader. The first ones are mainly used for tracking systems, where the precise location of a transponder is important. Usually, the beacon signal is detected by multiple readers at the same time, what permits locate the transponder. The second ones are widely used in access control (e.g. toll payment and checkpoint control). On the other hand, *passive transponders* have neither power source nor transmitter. They are powered only with energy provided by the reader. The radio signal sent by the reader is received by its antenna (coupling element). The transponder uses the energy received from this signal to power the microchip and exchange of data. The physical principles used for this data exchange depends on the

[3] A continuous or periodic radio signal, with limited information content, used to inform about some aspects of the state of an object such as location, identification or received signal strength.

frequency operation of the system. They are cheaper than active tag and do not required maintenance at all when they are deployed. However, the range of actuation and the computation power of the microchip are smaller. Transponders can also be classified also according to its complexity. One feature that affects in the complexity of a transponder is the capability of program its internal chip and modifies its memory. We can divide transponder in three classes: *read-only*, *write-once-read-many* (WORM) and *read-write*. In the read-only transponders data can be programmed during the manufacturing process and cannot be modify later by any mean. Usually, the data stored is the unique serial number of the transponder. They are used in supply chain management. WORM transponder are quite similar, but usually is the user and not the manufacturer who writes the content in the transponder. Practically, a WORM memory can be written a hundred of times and a common use case is access control. Finally, the read-write transponders use EEPROM or a Flash memory which permits from 100.000 to 1.000.000 reprogramming cycles. These devices are more expensive to produce, have higher power consumption and have higher security risks (tag tampering). This is the technology chosen to build NFC tags.

A *reader* is the device in charge of reading data from a transponder and/or writing new data on it. A reader typically contains a radio frequency module, a control unit and a coupling element. The radio frequency module also known as RF Transceiver provides the RF energy to activate and power the transponder. It controls and modulates the radio signal and filter, amplify and demodulate the signal received from the transponder. Some readers have two coupling elements (one for transmitting and one for receiving) while others use the same antenna for signal transmission and reception. Once the signal is demodulated, it is decoded by the control unit which can process the data further. Transponders are activated when they are in the interrogation area of the reader (that is, inside the space in which the transponder receives the radio signal from the reader with enough power to actuate). If the transponder needs power to enter in the active mode (passive transponders), it is supplied by the reader by means of its coupling units. Furthermore, the clock signal which controls the transponder's microprocessor is also supplied by the reader. Data transmission (full duplex, half duplex or simplex, depending on the RFID technology used) occurs once the transponder is powered and activated.

The *processing device* is the interface between the reader and the users. They operate on readers by sending commands to the processing device. It processes the data received from the reader and communicates the results to the user. There are multiple physical implementations of a processing device: a personal computer, a laptop, a smart-phone ... The communication link between the reader and processing device can use a wide variety of technologies both wireless (e.g. Bluetooth, Wi-Fi, or Zigbee) or wired (USB, Ethernet). Some literature does not consider the processing device as part of the RFID system. However, as shown by Schraberreiter et al. [17] it is an important part of the system and should be taken into account when analyzing, for example, the security of the whole system.

Another implementation of RFID technology is *Smart cards*: electronic data storage systems that are able to transmit and process the data. For convenience they are commonly incorporated into plastic cards. Nowadays they are used for IDs, credit cards, prepaid cards, e-tickets, and digital TV card for set-top-box receivers. Smart cards can be embedded in many different houses such as passports, mobile phones (SIM cards). Some authors consider that Smart Tokens is a better name for the devices that do not have appearance of a plastic card, though. Mayes and Markantonakis [18]

defines a set of criteria that a smart card must meet: a smart card participates in an automated electronic transaction, in which the card is used primarily to add security and it cannot be easily forged or copied. To that end, a smart card must store data securely and can host a range of algorithms and functions. The first smart cards used in large quantities were memory cards used as prepaid cards for phone calls or public transportation or as loyalty cards to store clients' points. These cards contain a very basic security logic that makes possible to protect the data against undesired manipulation. However, they cannot process modern cryptographic algorithms. To host cryptographic algorithms a card needs a microprocessor and a specialized operating system such as Multos, GlobalPlatform and JavaCard [19]. The last one has a special importance since it is also used in NFC. All of them permit storing and executing secure applications in the card. Traditionally, the smart cards must be inserted in a card host or reader. The smart card has eight contact points (including ground reference, power line, communication line and clock) that serve as serial communication interface between the reader and the card (Figure 3). Recent years have seen the apparition of contactless smart cards where the card and the reader are communicated through an RF channel. The main difference between wireless smart cards and current RFID tags is that the first ones can host an operating system and applications and the second cannot. In this Master Thesis we are only interested in contactless smart card due to its inclusion in NFC technology. Since RFID and contactless smart card share technology and protocols I am not going to go deeper in the smart card technology. Readers interested in this topic can check Rankle book [20].

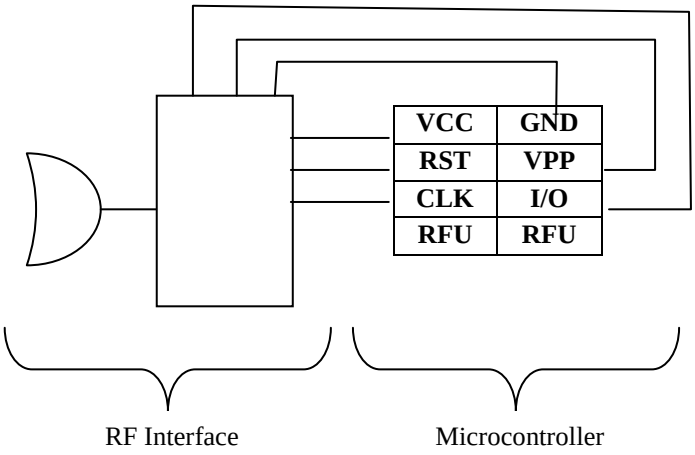


Figure 3. Contactless Smart Card Structure. Source: [15, p. 6].

2.1.2 Technology and physical principles

Operating Frequency

RFID operating frequencies range from 125 KHz till 5.8 GHz. RFID systems use four different frequency bands: LF (low-frequency, 30-300khz), HF (high-frequency, 3-30 MHz), UHF (Ultra High Frequency, 300 MHZ - 3GHZ) and Microwave (> 3GHz). High frequencies RFID systems do not work in the presence of water, non-conductive substances or metals. The absorption rate for water is higher in UHF and microwave. That is the reason why low frequency RFID systems are used when the signal has to

penetrate through objects or animal tissues. Furthermore, radio waves do not bounce on metals. In contrast, microwave systems permits broader interrogation area and are less sensible to interference caused by electromagnetic fields but consume a big amount of energy (majority of microwave systems needs active transponders).

Table 1. Main features of different RFID technologies. (Inspired by Laran RFID [21])

	LF	**HF**	**UHF**	**Microwave**
Typical frequencies	<135 KHz	13.56 MHz	860, 930, 950 MHz	2.45GHz
Typical Range	Several cm	1m	5m 100m (active tags)	1m 20m (active tags)
Coupling Methods	Inductive	Inductive, Magnetic / Capacitive (short distances)	Backscatter	Backscatter, Surface acoustic wave transponder.
Transponder type	Passive	Passive	Passive and active	Passive and active

Coupling methods

The coupling method defines the physical principles used to create the RF link between reader and transponder. The most common coupling methods in RFID are:

Inductive coupling. The inductive coupling is a physical phenomenon where one device transfer energy (voltage) to another through a variable magnetic field. This phenomenon takes place, for example, in a voltage transformer. In the RFID case the primary coil is in the reader and the secondary coil is in the transponder. RFID readers use the electromagnetic induction to provide energy to the transponder's chip. When the reader and the transponder are close enough, a variable current in the reader's coil generates a variable magnetic field (H). The magnetic field passes through the coil in the transponder generating a current flow (and hence a voltage if the coil has a load) that is proportional to the current provided in the reader. Figure 4 shows an equivalent circuit for the reader-transponder system. To increase the current in the reader and hence the magnetic field strength in the reader's coil a parallel resonant circuit is used. The inductance Lr and the capacitor Cr form a resonant circuit which delivers maximum power to its resonant frequency which matches the working frequency of the reader. The same strategy is used in the transponder, to get the maximum power. The Lt and Ct1 are chosen so they form a resonant circuit tuned to the reader's working frequency. The voltage Ut, that powers the chip, is proportional to the signal frequency, the number of windings of the coil and its area. It means that higher frequency requires less number of windings. If the reader is working to 13.56 MHz (NFC) the number of windings needed in the transponder coal's is less than 10, while to 125 KHz the number of windings is between 100 and 1000.

To transfer data from the reader to the transponder (forward channel) the digital signal is modulated using any digital modulation technique. The most common ones are ASK (Amplitude-shift keying), FSK (frequency-shift keying) and PSK (phase-shift keying). The chip in the transponder demodulates the signal and process the information.

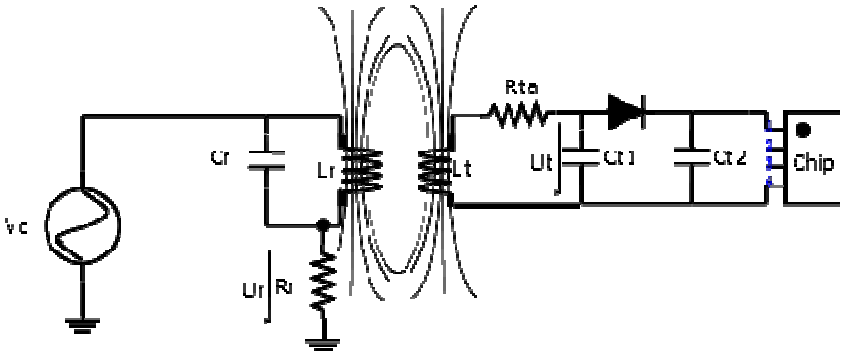


Figure 4. Reader-Transponder System equivalent circuit. Source: [15, Ch. 4].

The data transfer between the transponder and the reader (backward channel) is more complex since the transponder is a passive device. Hence, the data must be modulated using the same carrier as the one received by the reader. The most popular method to perform this modulation and send back data to the reader is named load modulation. If we alter the impedance in the transponder we alter the resonant circuit, and so we alter the voltage Ut in the transponder antenna. This also modifies the voltage in the inductance Lr, and so the voltage in the reader Ur. The variation of this voltage is the signal that the transponder wants to send to the reader. Opening and closing the switch S in Figure 5 produces a variation in the transponder impedance and hence in the voltage on the reader antenna and consequently on the reader resistor. The switch is controlled by the data signal that the transponder sends to the reader.

The coupling between the reader antenna and the transponder is very weak. So, the voltage fluctuations at the reader antenna are very small so the circuitry to detect the signal becomes very expensive. To cope with this problem RFID transponder generally use load modulation with subcarrier, that permits better filtering of the received signal, making circuitry cheaper. The main disadvantage of this method is that transmitting a subcarrier makes necessary a larger bandwidth.

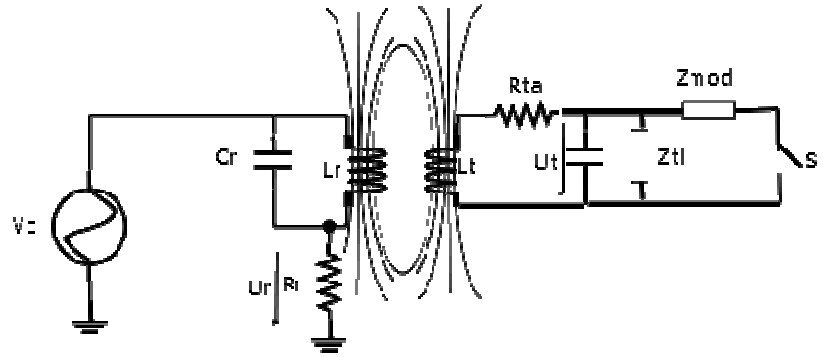


Figure 5. Reader-Transponder System equivalent circuit with load modulation. Source: [15, Ch. 4].

It is important to note that inductive coupling is only valid when the transponder is placed in the near field area of the reader, so it meets that $d <<< \lambda_{fc} * \frac{1}{2\Pi}$ where *d* is

the distance between the reader and the transponder and λ_{fc} is the wavelength of the reader's working frequency. In the near field, we just consider the magnetic field generated in the conductor. At longer distances, the magnetic field propagates and electric field developed by induction. Magnetic and electric fields feed each other increasingly to generate and electromagnetic wave. The area in which an electromagnetic wave is completely generated is named "far field". In the far field the transponder cannot send back feedback to the reader, since there is no inductive coupling. To a frequency of 13.56MHz, the theoretical maximum working distance for a reader is 22.1 cm.

Electromagnetic backscatter coupling: This method is used by RFID systems in which the range is longer than 1m and the operational frequency is in the UHF band or microwave. In this case the coupling is performed in the far field, and thus uses electromagnetic wave theory instead of magnetic induction. The physical working principles for this method are the same as the ones in radar technology: all electromagnetic waves are reflected by objects which are bigger than half of the wave's wavelength. If the object is in resonance with the wave (e.g. an antenna with the appropriate shape and size) the reflection efficiency is very big. The characteristics of the reflected wave depend, among other things, on the load of the antenna where the reflection happens. So, to transmit data from a transponder to a reader, it is only needed to change the value of the load resistor of the antenna in time with the data to transmit. The reflected wave is received by the reader where is processed. The main drawback is that the power received by the transponder is not always enough to feed the chip, so usually those transponders need a battery. Other disadvantage is that these systems are very prone to interference.

Capacitive coupling: This method is used for short range systems (less than 1 cm), in which the transponder is inserted or placed onto a reader. With capacitive coupling the reader and the transponder acts as if they were plates of a capacitor. The energy generated using this method is insufficient to power a microchip so it is combined with the inductive coupling. Data is transmitted using the capacitance coupling while the chip is powered using inductive coupling.

Magnetic coupling: This method is used in the same circumstances as capacitive coupling and shares the similar physical principles as inductive coupling. Since the distance between the reader and transponder is very short the efficiency is much higher. It is used for transponder with high power consumption.

Interrogation area range

It is highly correlated to the operating frequency and the coupling methods used. RFID systems can be divided in *close coupling systems* (distance between reader and transponders is less than 1 cm), *remote couples systems* (distances varies between 1 cm and 1m) and *long range system* (more than 1m and less than 100m [22]). Each application has an ideal range of actuation depending on the positional accuracy of the transponder, the minimum distance between transponders in practical operations and the time that the transponder stays in the interrogation area. It is very important to analyze the application and select the technology which provides the interrogation area range that more suits to its necessities.

2.1.3 Standards

The widely usage of RFID and Smart Cards in industry has led to a vast standardization process. Table 2 summarizes the main standards used in RFID and NFC.

Table 2. Main standards related to RFID

Number	Name
VDI 4470	Anti-theft Systems for Goods
ISO 7810	Physical characteristics of ID cards
ISO 7816 (1 to 9)	Smart card standard. Identification cards. Integrated circuit with Contacts
ISO 10536 (1 to 4)	Identification Cards – Contactless Integrated Circuit(s) Cards. Clouse-Coupling cards.
ISO 10374	Freight Containers - Automatic identification
ISO 11784, ISO 11785 and ISO 14223	Radio-frequency identification of animals. Code Structure
ISO 14443 (1 to 4)	Identification Cards – Proximity Integrated Circuit Cards
ISO 15693 (1 to 3)	Identification Cards – Contactless Integrated Circuit Cards and Vicinity Cards
ISO 15961	Information technology -- Radio frequency identification (RFID) for item management -- Data protocol: application interface
ISO 15962	Information technology -- Radio frequency identification (RFID) for item management - Data protocol: data encoding rules and logical memory functions
ISO 15963	Radio frequency identification for item management - Unique identification for RF tags
ISO 18000 (1 to 7) and ISO 18001	RFID for Item Management
ISO 18092, ECMA-340	Near Field Communication Interface and Protocol (NFCIP-1)
ISO 21481, ECMA-352	Near Field Communication Interface and Protocol (NFCIP-2)

2.2 NFC technology

2.2.1 Introduction to NFC

Near Field Communication (NFC) is a short range wireless technology which enables communication between two electronic devices whenever both are brought close together. Theoretically, communication range is around 10 cm. NFC operates on 13.56 MHz and supports data rates of 106, 212 and 424 Kbit/s. Higher speeds are theoretically possible [23], though. NFC is standardized in the ECMA-340 and ISO/IEC 18092. NFC incorporates the ISO/IEC 14443 both Type A and B, ISO/IEC 15693 and Felica (JIS-X 6319) standards, making it technological compatible with a wide set of smart cards and RFID tags currently in the market. Actually, NFC enhances classical communication between a reader and a RFID tag, providing the possibility of establish a bidirectional communication between two NFC enabled devices that are brought close together (peer-to-peer). NFC standardization process is led by the NFC Forum[4], a not-for-profit industry organization, founded on 2004 by Nokia, Sony and Philips. Apart from promoting standards defined by international organizations (ISO/IEC, ETSI and ECMA), NFC Forum defines its own technical specifications to assure interoperability between different devices in the NFC ecosystem.

NFC enables an intuitive, simple and secure communication between two NFC enabled devices. It is intuitive in the sense that communication starts by touching or bringing close together two NFC enabled devices. It is safe in two different ways: Firstly, NFC supports digital signature for authentication and communication encryption (NFC-SEC). Secondly, since both devices must be very close to establish a communication (several centimeters), it is quite unlikely that a communication starts without user explicit consent.

A NFC system is composed by two different elements namely the Initiator and the Target. The Initiator is the device which starts and directs the communication while the target responds to the initiator requests. In RFID systems the initiator is the reader while the target is the transponder or RFID tag. NFC protocol defines two communication modes namely passive and active. In active mode both the Initiator and the Target use their own RF field to enable the communication. In this mode the target responds to the initiator using its own RF field. In contrast, in the passive mode, the target does not create any RF field and the data is transmitted from the Target to the Initiator using a load modulation scheme in which the RF field is generated by the Initiator.

NFC devices support three different operation modes for its applications:

- **Reader/Writer mode.** In this mode a NFC device is capable of reading/writing RFID tags. NFC Forum defines which types of tags must be supported by any NFC device. The physical layer is compliant with the ISO/IEC 14443 and Felica schemes (JIS-X 6319). Basically, this mode converts the NFC device in a RFID reader.
- **Peer-to-Peer mode.** In this mode, two devices create a bidirectional communication to exchange data following the NFCIP-1 interface and protocol. This mode enables NFC technology as a clear competitor of Bluetooth technology when the communication does not require high data rate (for example, short data to transmit).

[4] www.nfc-forum.org

- **Card emulation mode.** In this mode the NFC device is seen as a traditional contactless smart card by an external reader. It enables payment and ticketing applications. The RF interface is defined in the ISO/IEC 14443, which is a common standard for existing payment infrastructure.

It is difficult to envision NFC as a technology itself but as a technology enabler instead. It must be used in combination with other technologies to create a usable application. NFC allows communication between two devices in a simple and safe way. And most important, since NFC can read/write RFID tags and they can be attached to multiple objects, NFC permits interaction with the environment using our mobile phone as mediator. Nowadays the vast majority of the applications fall in any of the following categories:

1. Service initiation. A NFC tag contains information about an application that is launched in the mobile phone when that tag is touched. The tag also contains a list of arguments and/or other configuration parameters that the application receives on start. They provide context information to the application. The information could be encoded in multiple formats such as plain text, formatted text (JSON or XML), or as an URL. The tag could also contain a phone number and possibly a text message to initiate a phone call or sending a SMS to that number when the tag is touched.
2. Connectivity. NFC is used to transmit information between two mobile devices when using the peer-to-peer mode. However, the amount of information to transmit should be relatively small taking into account that the maximum speed accepted in current NFC standards is 424 kbps. If the amount of information to transmit is big, NFC can be used to configure a wireless connection using another technology (Bluetooth, Wi-Fi...). In this case, NFC protocol is used to transmit to both pairs the settings to initiate the connection: in the case of Wi-Fi the name of the network and the WEP / WPA password while in the case of Bluetooth the address of the devices and the corresponding pins. The user does not insert any data manually since the devices configure themselves. In the case of Bluetooth it has been shown that it reduces considerably the handshake time [24].
3. Payment. The smart card emulation mode permits the use of a NFC enabled mobile phone as a credit card. Like any other smart card, the NFC phone has a secure area in which a bank can store the user's credentials and the credit card information. The same phone can store information of multiple credit cards from different banks and multiple secure payment applications. Paying for goods is as easy as holding the phone close to the payment terminal and inserting a pin to confirm the operation. Payment and ticketing are supposed to be the killer application which must boost NFC development. However, a conflict of interest between different actors in the NFC ecosystem (banks, credit card issuers, mobile phone operators and phone manufacturers), have make impossible, up to now, the success of commercial payment application using this technology.
4. Transit and Ticketing. RFID technology has been used in transit applications for long time, (e.g. for paying tolls). NFC transit and ticketing applications use the smart card emulation mode to store the ticket information on the mobile phone. The phone can be used as a prepaid card to store bus tickets, concert tickets, sport event tickets or even money to pay a toll. Tickets can be bought/refill online. Users can check their balance on their mobile phone screen. As in the case of payment,

multiple ticketing applications can be stored in the secure element of the mobile device.

5. Sharing information. Either the peer-to-peer mode or the read tag mode can be used to retrieve and/or share information. NFC tags or other mobile devices may act as data repository. In the same way, data stored in someone's phone can be transmitted to others' mobile devices using the peer-to-peer mode. For example, a NFC tag might contain the information of a product, a train timetable or the business card of a person. Users can store information in a NFC tag (writing the tag) or retrieving information from the tag (reading the tag). On the other hand, a person might have his/her own business card stored in his mobile phone and can share it to other person using the peer-to-peer mode.
6. Advertising. This is a particular case of sharing information and ticketing. However, due to its business potential, it worth a special mention. NFC tags can be placed in posters or close to retail displays. When the tag is touched users could watch video advertisements of different products, access to more specific product information, or getting discount tickets to buy a product. The advantage of NFC over other ubiquitous advertising methods (such as broadcasting Bluetooth advertisements) is that it is not intrusive. The user decides by touching the tag that he wants to receive the information.
7. Social network bookmarking. NFC permits interactivity with social networks in the real world. Users can update its state in a social network touching a NFC tag located in a particular place. Although, social networks can get position information from other sensors (e.g. GPS or Bluetooth) or asking the user, NFC simplifies the process and protects user privacy. To send information about the user location she/he touches a NFC tag instead of writing this information using the mobile phone application's interface. The user is not continuously tracked as it happens with other solutions such as GPS or Bluetooth. The user authorizes explicitly that the application knows his/her current position by touching the NFC tag. Another function is to add people who you meet to your social network. Bringing the two people's phone close together, they can add each other as friends in the social network.
8. Action trigger. This concept is quite related the creation of links between real and digital worlds by means of physical browsing. Touching tags embedded in the user environment is a trigger to modify the current state of an object or a set of objects. For example, NFC tags can be used as switches to turn on/off lights, TV set or to start the reproduction of a specific film in a nearby computer.

The number of NFC mobile phones is gradually increasing. Furthermore, in Asia, Felica Networks introduced their own short-range contactless technology in their phones named Felica. However, although this technology is supported by the NFC standards, Felica phones cannot communicate with any NFC device but only with other Felica compliant device. Nowadays all major mobile phone manufacturers have several NFC enabled phones in the market. For example, all Windows 8 and Google Nexus branded phones and tablets support NFC. The future of this technology is quite promising, although it is developing much slower than expected. Partners forming the NFC ecosystem are waiting for others to give "the" important step to start a mass production of devices and applications for NFC. Simon Pugh summarized very well this problem in 2007 [25]:

"We're in a bit of a chicken and egg situation at the moment, to be honest. The mobile operators are deciding when to ask the manufactures to build commercial quantities of the products. The manufacturers are sitting, waiting for commercial orders. The banks are saying: 'we're keen on doing this but we don't see them commercially available yet.' So all of the entities are sort of circling waiting for someone to move."

2.2.2 General Architecture

Figure 6 shows a general overview of the NFC technology architecture. All the three application modes (card emulation, peer-to-peer and reader/writer) share the same physical layer. The three modes transmit data to a single application.

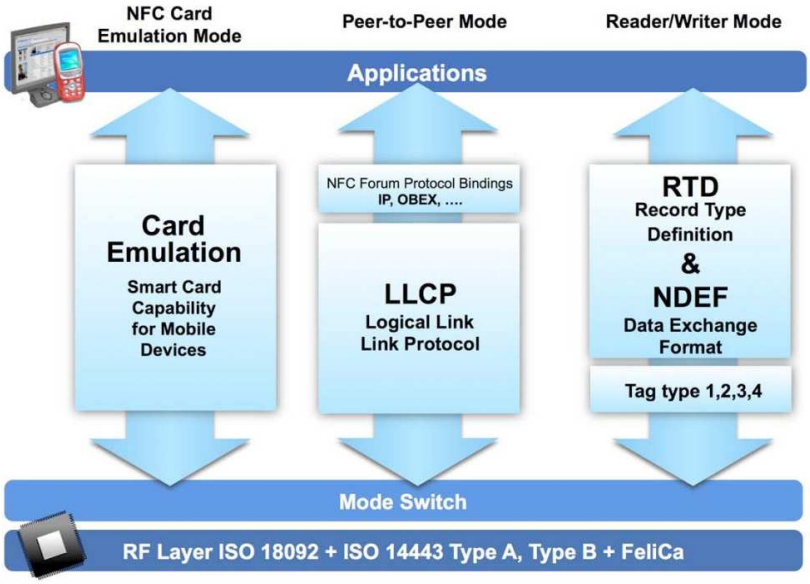


Figure 6. NFC general architecture. Copyright NFC Forum. Source: [26]

The following subsections explain briefly NFC main standards and NFC Forum Technical Specifications. In the following subchapters I emphasize the most important aspects of the standards and technical specifications. For more info download the corresponding document[5].

2.2.3 RF Layer Protocols

NFC utilizes the ISO/IEC 14443, one of the most used standards in the contactless chip card communication industry, for its RF layer. This fact makes NFC compatible with the majority of RFID tags working at 13.56 MHz and with multiple contactless chip card readers. NFC has defined its own standard (ISO/IEC 18092 or ECMA 340) for the peer-to-peer mode. It uses the lowest layers of the ISO 14443. Figure 7 summarizes the main standards and technical specifications used in the NFC RF Layer.

[5] ECMA standards can be download from http://www.ecma-international.org/publications/ , ISO/IEC from http://www.iso.org/iso/home/standards.htm (they are not free) and NFC Forum Technical specifications from http://www.nfc-forum.org/specs/

ISO 14443

This standard is one of the most used in contactless cards. For example, Mifare products (using NXP chip) and Felica follows this standard. The range of operation is of 10 cm. It consists of four different parts: (1) Physical characteristics, (2) Radio Frequency power and signal interfaces, (3) Initialization and anti-collision, (4) Transmission protocol. As shown in Figure 8 the second and third layer defines two different versions namely Type A and Type B. The layer two and three of the standard are of special importance since it is used also in the NFCIP-1. The standard is summarized in the following pages.

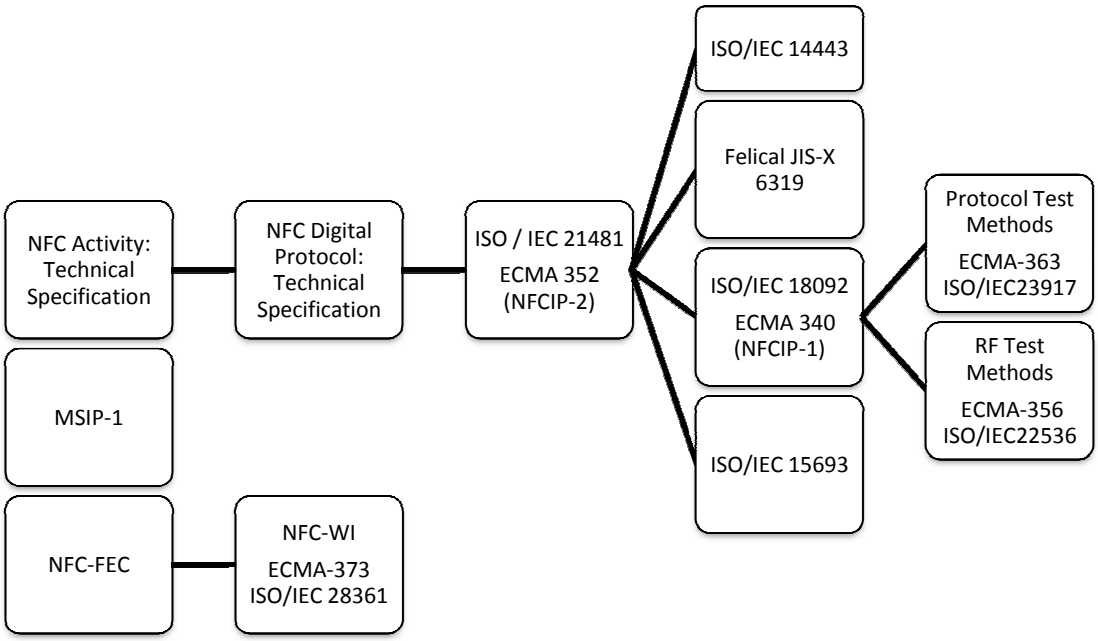


Figure 7. NFC related standards and technical specifications for RF Layer. Source: Standard ECMA-340.[6]

Figure 8. ISO 14443 protocol stack. Source: Standard ISO/IEC 14443, Proximity cards (PICCs).[7]

[6] http://www.ecma-international.org/publications/standards/Ecma-340.htm
[7] http://wg8.de/sd1.html#14443

Physical characteristics: Defines the physical characteristics of the PICC or transponder including the minimum level of electromagnetic field that a card must support without damage. It also defines other factors such as the size of the token, tolerance with regard to ultraviolet rays, X-rays … This part is not used by any NFC standard or recommendation.

Radio frequency and signal interface: The reader produces an alternating magnetic field with a frequency of 13.56 MHz assuring a magnetic field within the range of 1.5 A/m < H <7.5 A/m. This field permits bidirectional communication between reader and transponder. Basic data rate is 106 Kbps. However, some readers and transponders support data rates of 212, 424 and 848 kbps. Table 3 summarizes the encoding and modulation used for different channels. Note that backward channel uses always load modulation with a subcarrier of 847 KHz.

Table 3. Encodings and modulation for different channels. Forward channel is the channel between reader and transponder while backward channel is the communication channel between transponder and reader. Source: Standard ISO/IEC 14443-2.[8]

	Type A	**Type A**	**Type B**	**Type B**
Channel	Forward channel	Backward channel	Forward channel	Backward channel
Binary Encoding	Modified Miller	Manchester	NRZ	NRZ
Modulation	100% ASK modulated onto carrier (13.56 MHz)1	100% ASK modulated onto a 847 KHz subcarrier.2	10% ASK modulated onto carrier (13.56 MHz)	BPSK modulated onto an 847 KHz subcarrier.

1 Since 2005 for speeds higher to 106kbps 80% ASK, 60%ASK and 40%ASK were added

2 Since 2005 for speeds higher to 106kbps the binary encoding is NRZ and the Modulation is BPSK instead of ASK.

Initialization and anti-collision: Defines the data frame format and the commands used to detect and initialize communication with a token. Detection is done by polling the interrogation area. Anti-collision is used in situation in which several transponders are detected simultaneously and the reader is already communicating with one transponder, to avoid interference.

The Type A data frame is delimited by a Start-Of_Frame (SOF) and End-Of-Frame (EOF) symbols. Error detection and correction consists on a reserved parity bit for each byte of data and a 2 byte CRC sent at the end of the frame. During initialization the transponder acts as a state machine. Figure 9 shows this state diagram of a transponder. When a field is detected the transponder is moved to the IDLE state. The transponder stays in that state until it receives a REQA command that is sent periodically by the reader to detect transponders in the reading area. The transponder responds with a ATQA frame and changes to READY state. It is possible that multiple tokens are in the READY state at the same time, so an anti-collision algorithm is needed to continue transmission. Detecting a collision is simple when using Manchester encoding. In Manchester encoding 0 are represented as low to high transition and 1 as high to low transition. The transitions always occur at the midpoint

[8] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50941

of the period. When the voltage is high during the whole period, there is a collision. However, for collision detection all frames must be synchronized. The whole anti-collision algorithm is quite complex and can be consult from the standard. Type A uses a binary search tree algorithm to activate a transponder. Every transponder has a unique serial number of 4, 7 or 10 bytes. After the anti-collision loop ends, the reader knows the serial number of one of the transponders, to which it sends a SELECT command. The transponder responds with a SAK (SELECT Acknowledge) message and moves to the ACTIVE state. Before starting the transmission of application specific messages the reader responds with a RATS (Request and Answer to Select) in which reader send some parameters to setup the communication such as CID (unique number associated to this transponder). From ACTIVE state the transponder can be moved to HALT state. The HALT command can be sent either the reader or by higher protocol layers. The transponder can move again to the READY state via a WAKE-UP.

The data frame format and state diagram of Type B protocol is quite similar to the one defined for Type A. Type B transponders includes an Application Family Identifier. The REQB command contains an AFI field. Only the transponders which AFI is the same as the one sent in the REQB command can continue with the anti-collision process. The anti-collision algorithm is also different for the one defined in Type A. In this case it uses slotted ALOHA procedure.

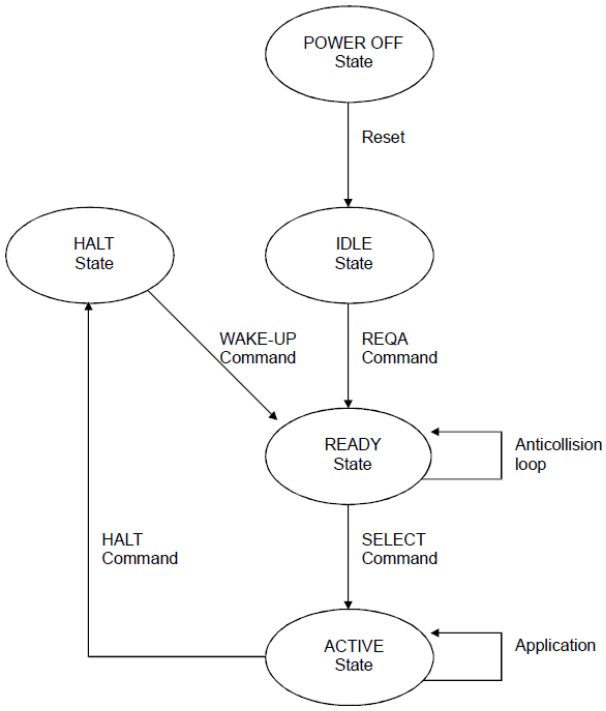


Figure 9. ISO 14443 transponder state diagram for type A. Source: Standard ISO/IEC 14443-2.[9]

Command protocol: Once the initialization of the transponder has ended successfully, the reader and the transponder can exchange application data. A previous setup is required to agree baud rate, frame size... The commands to be sent from the transponder to the reader are encapsulated in APDUs (Application data Unit). The protocol defined in ISO 14443-4 is also known as T=CL since is heavily based on the T=1 (ISO 7816-3) used for contact smart cards. It is a block oriented protocol in which

[9] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50941

the communication is based on the master-slave paradigm. The master (reader) sends commands (APDUs) to the slave (transponder), which executes them and sends back a response. This protocol implements the transport and the application layer of the OSI structure, encapsulating the APDUs in frames. There are three different kinds of frames: (1) information blocks which transmit data (APDUs), (2) reception blocks which acknowledge the reception of another frame and (3) system blocks which exchange protocol control data. The structure of a frame is shown in Figure 10.

The PCB is a byte which defines the type of block that is going to be sent. The CID is a byte which allows identifying a single transponder, among the ones which are in the Active state. The NAD is a byte defined to create logical connections between the transponder and the terminal. It provides compatibility with the T=1 protocol. The INF is where the actual data is stored. The CRC is used for error detection and recovery.

PCB	[CID]	[NAD]	INF / APDU	CRC

Figure 10. Structure of a frame. Between [] the optional fields. Source: Standard ISO/IEC 14443-4.[10]

NFCIP-1: ISO 18092 - ECMA 340

As shown in Figure 11 the NFCIP protocol uses the same radio and initialization stack as the ISO 14443 but redefines the application protocol. It permits peer-to-peer communication between two NFC enabled devices via magnetic coupling. The standard defines passive and active communication modes between the initiator (the entity which starts the communication) and the target. In passive mode, the communication is similar as explained in previous section. In the active mode, both the initiator and the target use their own generated field to couple the other entity, and transmit data. Field generation is alternately. The carrier frequency is 13.56 MHZ, the magnetic field of both devices must be within the range of 1.5 A/m <H <7.5 A/m and the accepted speeds are 106, 212 and 424 kbps. The application must select the communication mode (passive or active) as well as the speed Table 4 summarizes modulation and encoding for different channels and communication modes.

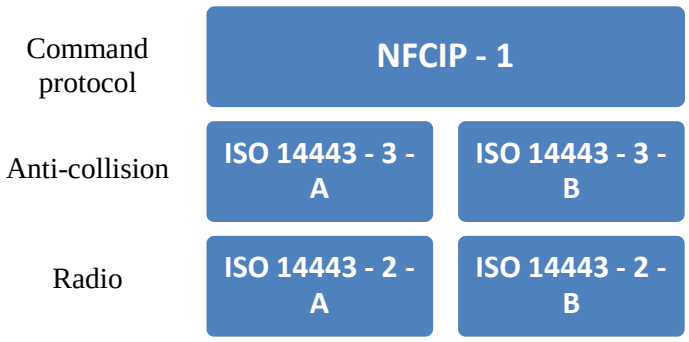


Figure 11. ISO 14443 protocol stack for NFC. Source: Standard ECMA-340.[11]

[10] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54823
[11] http://www.ecma-international.org/publications/standards/Ecma-340.htm

Table 4. Encodings and modulation for different channels. Forward channel is the channel between initiator and target while backward channel is opposite one. Source: Standard ECMA-340.[12]

	Active	**Active**	**Passive**	**Passive**	**Passive**	**Passive**
Channel	Both	Both	Forward channel	Backward channel	Forward channel	Backward channel
Speed	106 kbps	212 and 424 kbps	106 kbps	106 kbps	212 and 424 kbps	212 and 424 kbps
Binary Encoding	Modified Miller	Manchester	Modified Miller	Manchester	Manchester	Manchester
Modulation	100% ASK modulated onto carrier (13.56 MHz)	8-30% ASK modulated onto carrier (13.56 MHz)	100% ASK modulated onto carrier (13.56 MHz)	100% ASK modulated onto 847 KHz subcarrier.	8-30% ASK modulated onto carrier (13.56 MHz)	100% ASK modulated onto carrier (13.56 MHz)

The communication process is divided in two parts namely Initialization and Transport.

Initialization. The Initiator cannot create its own RF Field if it detects another RF field, to avoid collisions. Thus, the initiator senses continuously the environment. Furthermore, in active communication mode, the target cannot generate a RF field when another already exists. The Initialization protocol is different depending on the communication mode and speed:

- *Passive communication at 106 kbps*. Follows similar initialization and collision avoidance techniques as ISO 14443-3A standard modifying the names of some commands and states. E.g.: IDLE state is named SENSE state, READY state is named RESOLUTION state while ACTIVE state is named SELECTED state. The UID is substituted by the NFCID. The collision avoidance loop is named SDD (Single Device Detection). After the SDD, the initiator and target share the communication preferences using commands of type PSL. Figure 13 shows the frame format defined in the standard. If the payload has more than one byte, then each byte must include a parity bit in the less significant position.

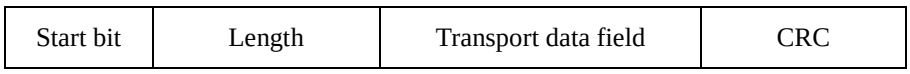


Figure 12. Frame structure defined in ISO 18092 and ECMA-340 for passive communications and speeds of 212 and 424 kbps. Source: Standard ISO/IEC 18092 (NFCIP-1).[13]

- *Passive communication at 212 and 424 kbps*. It implements the collision avoidance loop (SDD) in a similar way as ISO 14443-3B. In this case the UID is named NFCID2, and it is an 8 byte number. After the SDD has ended, the initiator and target share the communication preferences using PSL commands. Figure 15 shows a frame as it is defined in the standard. The Length field indicates the

[12] http://www.ecma-international.org/publications/standards/Ecma-340.htm
[13] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56692

number of bytes of the payload plus one. The sync field contains a fixed number (#B24D) specified in the standard.

Preamble	Sync	Length	Transport data field	CRC

Figure 13. Frame structure defined in ISO 18092 and ECMA-340 for passive communications and speeds of 212 and 424 kbps. Source: Standard ISO/IEC 18092 (NFCIP-1).[14]

- *Active Communication mode*: In this mode the collision avoidance is much simpler. First the initiator checks that there is no electromagnetic field and sends and ATR_REQ command. After that switches off the field. The target checks that there is no field and answers with an ATR_RES. The Initiator and Target share set up parameters using PSL commands. Finally, the initiator sends a DEP_REQ to start the exchange protocol. The RF field is alternatively switching on and off to avoid collision between initiator and target.

Transport protocol. The transport protocol is divided in three parts: activation of the protocols (exchange of attributes and communication parameters selection), the data exchange protocol and the deactivation protocol. The activation protocol has been briefly explained in the initialization section. All the data to be transferred during the data exchange is stored in the transport data field (Figure 12 and Figure 13). The transport data field is shown in Figure 14. The bytes CMD0 and CMD1 are used to define the commands. PFB byte is used to convey the information required for controlling the transmission and for chaining different frames. It also defines the type of data sent namely: information, ACK/NACK or supervisory request/response. The voluntary DID field is used to store the id of the initiator or target. The voluntary NAD field is reserved as an identifier to create logical connections on the initiator and the target.

CMD0	CMD1	PFB	[DID]	[NAD]	Data bytes

Figure 14. Transport data field defined in ISO 18092 and ECMA-340. Source: Standard ISO/IEC 18092 (NFCIP-1).[14] above

NFCIP-2: ISO 21481 - ECMA 352

This standard specifies for ECMA-340, a communication mode selection mechanism for the ISO/IEC 18092 (NFC) and for the ISO/IEC 14443 (PICC or PCD) which is designed not to disturb possible ongoing communication at 13.56 MHz band. The mode selection algorithm works as follows:

1. If the NFCIP-2 device detects an external RF field it selects either the PICC mode or the NFC mode.
2. If the NFCIP-2 device does not detect an external RF field it shall select the NFC mode, or the PCD Mode. The device must first perform RF detection, initial RF generation and eventually enter in the selected mode.

[14] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56692

2.2.4 Upper layers' protocols

NFC Digital Protocol Technical Specification

This specification is sat on top of ISO 18092 and ISO 14443 standards. It harmonizes the digital interface and the half-duplex transmission protocol of both standards for the four roles that a device can have namely: initiator, target, reader/writer and card emulator. The goal is to assure interoperability between different NFC devices and with legacy platforms.

This standard defined three different technologies: NFC-A (based on ISO/IEC 14443-A), NFC-B (based on ISO/IEC 14443-B) and NFC-F (based on ISO 18092). They differ in the possible communication modes, bit rate, modulation scheme, bit encoding, frame format and command set. All of them use however the same carrier (13.56 MHz). Table 5 defines the main properties of these 3 technologies.

This standard defines two possible initial states for any device: Listen mode (those devices that do not generate carrier, and so, cannot start a communication) and Poll mode (those devices that generate a carrier and "polls" for other devices). NFC devices exchange data using half-duplex protocols (only one device send data at a time). A device in Poll mode cannot move to Listener mode until it receives a response from listener or timeout expires.

Table 5**.** Modulation, collision protocol and command set defined in NFC digital Protocol Technical Specification. Source: NFC Forum. NFC Digital Protocol Technical Specification.[15]

	NFC-A	**NFC-B**	**NFC-F**
Base standard	ISO 14443-A	ISO 14443-B	ISO 18092
Byte modulation and encoding	100% ASK with Modified Miller (forward) and OOK subcarrier modulation with Manchester coding (backward)	ASK with NRZ-L encoding (forward) and BPSK modulation in a subcarrier with NRZ-L encoding (backward)	ASK modulation with Manchester encoding.
Collision	Superposition of "0" and "1"	Superposition of multiple responses	Superposition of multiple responses
Command set	ALL_REQ, SENS_REQ SDD_REQ SEL_REQ SLP_REQ	ALL_REQB, SENS_REQB, SLOT_MARKER, SLPB_REQ	SENSF_REQ

NFC Activity Technical Specification

This standard explains how the NFC Digital Protocol Specification can be used to setup communication with another NFC device or NFC tag by means of Activities, that is, processes running on the device. An activity could be viewed as a flow or combination of flows that looks like a library function that a program may call. Activities combine the frames or block of data taken from the physical layer into functional block. Activities are combined in profiles. A profile has specific

[15] http://www.nfc-forum.org/specs/spec_list/

configuration parameters and covers a particular use case. A profile defines a sequence of activities to be performed by a NFC device. The Resolution Process is an algorithm controlled by the upper layer and is in charge of selecting the next Activity in the sequence, insert on it the corresponding input parameters and process the output. This standard defines the following profiles:

- P2P poll profile: Create a communication between two NFC devices using the peer-to-peer mode. The profile is in charge of selecting the maximum speed available given the technology.
- NDEF Poll Profile: It is used in the Reader/Writer mode to access/modify data on a tag. It first searches for a NFC tag containing data in NDEF format (section 2.2.5) and establish the communication with it. If multiple tags are detected the profile ends without creating any connection.
- P2PNDEF Poll Profile: The profile detects the kind of device in Listening Mode. If the device is a NFC tag with data in NDEF format it works as in NDEF Poll Profile, while if the detected devices is a device able to use NFC-DEP protocol it acts as P2P Poll profile.

2.2.5 Reader/Writer mode

The NFC Forum has defined a format of encapsulating data that is stored in a NFC Tag and that can be exchanged from/to a NFC device. Furthermore it defines four different tag types (based on existing standards), how NDEF messages must be stored in each type of tag and how a NFC enabled device must operate this data.

NDEF

NFC Data Exchange Format (NDEF) defines a binary message encapsulation format to store data in NFC tags and exchange information between NFC enabled devices. A message is formed by one or more records. Each record is an application specific data payload that can be processed by a NDEF application. A record structure is defined by its type. NDEF records can be chained together to support larger payloads. Furthermore a payload can be fragmented into chunks in different records. The NFC device should be able to assembly these chunks together to obtain the desired payload. An NDEF record contains 3 different main fields: payload length, payload type and payload identifier (to make cross reference on payloads). Figure 15 shows the structure of a NDEF record.

The NDEF standard defines four record types: NFC well-known types, Media type, Absolute URI and NFC Forum External type.

NFC Forum well-known types. The five well-known types defined by the NFC Forum are:

- *Text*. General purpose text field to add metadata to objects. Each record contains the ISO/IANA language code. If the same NDEF message contains multiple Text records, the NFC device selects the text in the device´s language.
- *URI*: Permits store URIs on a record. The protocol of the URI is encoded using just one byte.
- *Smart Poster*: It permits associate different kind of multimedia content (video, audio, webpage) to an object. The content is a NDEF message with the following records:

- Title record. It defines the name that the device associates to the content.
- Uri record. Uri of the multimedia content.
- Action Record. It specifies how the application must process the Uri content.
- Icon Record. An image which identifies the content of the Uri.
- Size Record. The size of the content defined in the Uri.
- Type record. The mime type of the object identified by the Uri record.

● *Generic Control*. A Generic Control Record is used to request a specific action to an NFC enabled device. The structure is a NDEF message which contains the following records:

- Target Record: Function to handle this record.
- Action Record: Specifies the requested action for the target function.
- Data Record: Contains the arguments to pass to the function.

● *Signature*. Used to sign single or multiple records and verify the authenticity and integrity of data stored in a NDEF message.

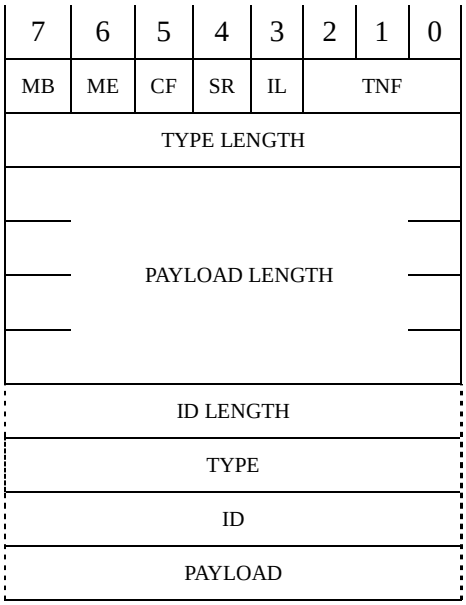


Figure 15. NDEF record structure. TNF defines the type of the record. The CF flag is used for fragmentation and assembly of records in chunks. Source: NFC Forum. NFC Record Type Definition (RTD) Technical Specification.[16]

Media type. The name of the type is a mime-type as defined in RFC 2046. The content is a file with that mime type.

Absolute URI. It contains an URI.

NFC Forum External type. Types defined by application developers.

Tag types

The NFC Forum defines four different types for NFC tags. Each type is based on an existing tag technology. The purpose of the specification is to define how a NFC enabled device detects, reads and write NDEF data into a NFC tag. Each type defines

[16] http://www.nfc-forum.org/specs/spec_dashboard/

also how the NFC tag memory must be used to store NDEF messages. The four types are numbered from 1 to 4.

NFC Forum Tag type 1. This type is commercially available with the Topaz brand. It is based in ISO14443A standard. Tags of this type are read and rewrite capable, while permits to configure the tag as read-only. The memory availability ranges between 96 bytes and 2kbytes. Communication speed is 106kbits/s. The memory is divided in blocks of 8 bytes each. A header ROM block (not readable using READ command) indicates the type of tag. The block 0x00 contains the UID of the tag. The block 0x01 contains what is called Capability Container, which indicates if the tag contains a NDEF message, the Version number of the tag 1 type, the physical memory size of the tag and if the Capability Container data can be rewritten. Data is stored in TLV block. A TLV block is formed by three fields: T (1 byte) identifies the type of the block, L (1 to 3 bytes) identifies the length of the value field while V contains the payload of the block.

NFC Forum Tag type 2. This type is commercially available as Mifare Ultralight (NXP). It has similar technical requirements as NFC Forum Tag type 1. The memory is divided in blocks of 4 bytes each. A sector contains 256 blocks. The block 0 is reserved for the UID, the block 1 is reserved for the serial number, the block 2 is reserved for manufacturer internal use while the block 3 is reserved for the capability container (which structure and functionality is similar to type 1 tags). The rest of the blocks are reserved for data storage. Data is stored in TLV block which structure is similar to type 1 tags. Commands accepted by this type of tags are READ block, WRITE block and SECTOR SELECT (it is necessary to select a sector before operating on it),

NFC Forum Tag type 3. This type is commercially available as FeliCa (Sony tag based on Japanese Industrial Standard X6319-4). Tags are preconfigured at manufacturer to be re-writable or read-only. It has a variable memory limit and a speed of either 212 kbps or 424 kbps. Memory is divided in blocks of 16 bytes. A service (similar concept than file in a file system) has reserved a number of memory blocks. Furthermore a type 3 tag contains some fields to store system information: Manufacturer ID, System Definition and Service Definition information present for each service in the tag. NDEF data must be stored using memory blocks assigned to the Service with code 0x000B. The first block of this service is the Attribute Information Block which has a purpose similar to the capability container blocks in type 1 and 2 tags. The commands accepted by this type of tags are Polling (initialize the communication), Check command (reads a number of blocks) and Update command (writes a number or blocks).

NFC Forum Tag type 4. This type is commercially available as DESfire (NXP). These tags are compatible with ISO 14443A and B standard. They are preconfigured at manufacture time to be either re-writable or read-only. Memory is up to 32 Kbytes and the communication speed is up to 424 kbps. The data is stored encapsulated in applications. The NDEF Application contains two files: the Capability Container file (similar to type 1 and 2 tags) and the NDEF file which contains the data. The Capability container file stores its data using the TLV blocks described for type 1 and 2 tags. The commands accepted by this tag are Select (to select an application or file), Read (to read data from file), Update (Erase and write data to file).

2.2.6 Peer-to-peer mode

NFC Logical Link Control Protocol

The Logical Link Control Protocol supports peer-to-peer communication between two NFC enabled devices over the protocols defined by the NFCIP-1. LLCP provides the following services:

- Link activation, supervision and deactivation: Defines how the connection between two NFC devices is established and released. It also monitors the connection. Serialize all connections PDU exchange.
- Connection oriented transport: Provides a reliable connection oriented protocol (similar to TCP). PDUs are numbered and ACK are sent back when a package has been received. A sliding windows controls the number of unacknowledged PDUs. A connection must be established before starting communication. This connection is kept during the whole communication.
- Connectionless transport: When upper layers implement their own flow control mechanism there is no need to keep a connection oriented approach. There is no ACK between receiver and sender. In this mode there is no need of connection establishment so it allows spontaneous exchange of data.
- Asynchronous balanced communication: Initialization, monitoring, error recovery and transmission of information can be done asynchronously.
- Protocol multiplexing: The same LLCP can offer service to multiple applications simultaneously.

The format of a LLC PDU is shown in Figure 16.

Destination Service AP	Payload type	Source Service AP	Sequence number	Payload

Figure 16. LLC PDU. The payload type is a 4 bits field which informs about the type of PDU: e.g. if it is a signaling PDU, information PDU, ACK PDU or errors PDUs. Source: NFC Logical Link Control Protocol (LLCP) Technical Specification.[17]

NFC Simple NDEF Exchange Protocol (SNEP)

LLCP supports several transport protocols. The simplest one is SNEP (Simple NDEF exchange protocol) which permits exchange of NDEF messages. Other protocols such OBEX and IP are also supported, but they need some specific bindings. The SNEP is a request/response protocol which exchange NDEF messages. Each request contains a version, a request method (continue, get, put or reject), the length of the information and the payload that (a NDEF message). The response message contains the following fields: Version, Response code (continue, success, not found, bad request, not implemented and reject), payload length and payload. SNEP accepts fragmentation and reassembly.

[17] http://www.nfc-forum.org/specs/spec_dashboard/

2.2.7 Smart card mode

Smart card mode is not used in this master thesis and its usage is far away from the research I am doing at this moment. I won't go further in technical details but give a shallow overview. In this mode NFC devices function as smart cards. One mobile device can store multiple smart cards applications in a single device. A secure element stores the application information. The secure element is an external element and is not part of the NFC specifications. The protocol used to communicate the secure element with the NFC chip is SWP (Single Wire Protocol). The HCI (Host Controlled Interface) is a logical interface on top of the SWP, allowing a NFC application to communicate with the secure element. Classical applications are payment, ticketing and access control.

2.2.8 Connection handover

One of the disadvantages of NFC communication is the low data rate it supports. In contrast, NFC makes the interaction faster since there is no need of a handshaking protocol or password insertion. Hence, NFC can be used as a mechanism to negotiate and activate an alternative communication carrier between two to transmit a big amount of data between two NFC devices. The NFC Forum has defined two different handover protocols: Negotiated handover and static handover. In the first case a handover requester offers several carriers alternatives to communicate with the handover selector device. The Handover Selector devices choose one carrier and communicate its selection to the Requester. In the second case, the Handover Selector is not a NFC enabled device but has a NFC tag attached. The NFC tag contains all the information to establish a connection with the Handover selector. The Requester and the Selector share two types of NDEF messages: Handover Request and Handover Select Record ("Hr" and "Hs").

2.2.9 Protocol stacks, programming environments and chips

The popularity of NFC technology has encouraged chips suppliers to develop NFC controllers' chips. Among them stands out NXP (with the PN53X and PN54X family), STMicroelectronics, Broadcom and Texas Instruments. Furthermore, since the NFC technology is based on ISO13334 standard, NFC is compatible with legacy RFID readers. Nowadays there are several NFC stacks that permits creating low level APIs for the NFC chips. Among the most known are:

- NXP FRI (NXP Forum Reference implementation): Open source protocol stack mainly targeted to NXP chips. This stack permits to manage the chip using Host Controller Interface commands. It supports all modes defined by the NFC standard and includes SWP for card emulation. It is fully written in C and has been ported to Android (using JNI).
- NFC-Open: Open source protocol stack promoted by Inside Secure. It supports all working modes defined by the NFC Forum including support for accessing to external secure element by means of a Security Stack. It is a portable code for several platforms including Windows, Linux and Android.
- NFCStack+: Propietary protocol stack defined by Stollmann. It supports all the functioning modes defined by the NFC Forum. Right now it offers support just for STMicroelectronics, NXP and Broadcom chips. This stack is ready to be used in

Linux, Android, Windows and QNX. This stack is integrated with Bluetooth to provide easy pairing between two NFC devices.

- libnfc: Not exactly a NFC stack but a C open source library for accessing NXP chips. This library is claimed to work in any POSIX-system as well as in Windows.

Furthermore, there exists a high level J2ME API standard namely JSR257. This API was released in 2006 and describe the interfaces to enable contactless transactions between a NFC enabled J2ME mobile phone and other NFC enabled device. Access to the secure element is not included in this JSR but in the JSR177. Currently, since the main mobile platforms that support NFC technology does not use J2ME, this API has fall in disuse.

2.3 Security issues

There are multiple attack surfaces in a NFC system. Attackers might, for example, perform attacks to some parts of the NFC system such as the backend system or to the device which hosts the NFC reader (Host Controller). In this section I am treating only possible attacks derived from using NFC technology. NFC is heavily based on RFID; any attack targeted to RFID can be extrapolated to NFC both for the Read/Write mode and for the P2P mode. Some possible attacks can be found from [27].

The wireless link is not encrypted so NFC is exposed to a wide set of attacks range from relay attack [28] or man-in-the-middle-attack to eavesdropping and data modification, insertion and corruption (DoS attack). Man-in-the-middle-attack is highly difficult to perform due to the fact than in the NFC link is only possible to have one device transmitting through the RF link simultaneously and NFC range is very short [29]. Data insertion is only possible when the answering device takes too long to response. Data modification, however, is somehow possible depending on the modulation scheme used. Data corruption and eavesdropping are quite easy to perform.

Besides attacks to the RF Link, NFC is also prone to attack to the devices. When using the Reader/Writer mode there are two attack targets: the mobile phone and the application itself. Data on the tag can be tampered. It is also possible to physically substitute the target for a tag created by the attacker or even sticking one tag on top of another and making the original one unusable (sending for example a high power electromagnetic pulse). By tampering the target the system is open to multiple types of attacks such as malicious code insertion, data spoofing or phising attack [30].

Readers placed in the environment are also attack targets, e.g. destroying or removing the device. The last case is especially problematic since using back-engineering the attacker might get access to sensible data such as keys used to communicate with other external devices or even with the backend systems.

The ECMA has defined a security protocol to try to cope with the risks related to using unencrypted communication channel. The ECMA 385 (NFC-SEC) defines an encryption mechanism over NFCIP protocol to protect communication against eavesdropping and data modification. The data is encrypted and send through a secure channel. The keys are shared between both end points or can use public key cryptography (using ECDH key exchange scheme and AES encryption algorithm defined by ECMA 386. To protect data written on NFC tags NFC Forum published the NDEF signature record type technical specification. Signature records are placed in between other NDEF records inside a NDEF message. Each signature record signs all

preceding NDEF records in the NDEF message. A signature record consists of three different parts: version information, a digital signature and a certificate chain. Both the digital signature and the certificate chain can be written in the record as an URI which links to the real data. This protocol supports the following signature types: PKCS, PKCS with SHA-1, DSA and ECDSA. The NDEF records signed in this way just assure the authenticity of the Type, ID and payload fields of the record. Roland et al. [31] have identified several potential attacks to signed records exploiting three main weakness: the possibility of signing a set of NFC records and not the whole NDEF message, the storage of the record's signature and the certificate chain in an external server and the not signing the record header.

2.4 NFC applications and trials

The cities of Oulu (Finland) and Nice (France) have stood out in the deployment of NFC based solutions. A group of elders from the city of Oulu tested an application which permits them to choose the food that the elderly care personnel should bring home [32]. The interface was built by attaching NFC tags into a paper meal menu for the whole week. The night before, the elderly consults the menu and touches with his/her mobile phone the option that he/she desires. They did not use the phone's keypad at all. Furthermore, the personnel of the logistic company in charge of delivering the food used a NFC tag placed at the door of the elder's house to send a confirmation of the delivery to the central. Ervasti et al. [33] presents a system to control the attendance of 6-to-8 years old children to school. Each child was given an NFC enabled card. When the child enters in the classroom they had to touch a NFC reader placed on the class. In some groups the teacher scanned the card with a mobile phone instead. This information was sent to the parents via SMS. Another Oulu's school participated in a NFC pilot in 2008. School placed info posters augmented with NFC tags in their premises [34, pp. 110 – 114]. Touching the poster with their NFC phone, pupils had access to the daily schedule, list of homework. Pupils could also access to multimedia content through REACHeS. Alisto et al. [35] report other multiple pilots e.g. a parking fee payment application or an application to order food in a restaurant. I report in [36] an application built in the University of Oulu zoological museum. NFC tags were placed on the show windows exhibiting animals. Museum visitors obtained multimedia information about an animal by touching the associated tag. Siira and his partners [37] reports a location-based mobile wiki using NFC.

Nice is the first city with a real commercial NFC deployment. French Mobile Network Operators collaborated with the University of Nice to implement and deploy such commercial services [38]. Among them worth mentioning the Nice Future Campus which tries to replace physical students' ID card with a mobile phone. Students use the card to pay buses, loan books and send messages to other students. In the MBDS project, researchers augmented cultural places (e.g. a museum) with NFC tags. Using NFC tags as tracking system, tourists could walk through predefined cultural paths.

Although there are not many NFC successful business applications on the market yet, many other research prototypes have been deployed all around the world. University of Cordoba, for example, is studying how integrate everyday-life administrative applications into university environments using NFC as interaction technology [39]. Researchers from University of St. Gallen have proved NFC as good technology for patients to interact with Ambient Assisted Living systems at

home [40]. NFC has been used also to build ubiquitous games [41], [42] and a mobile sales assistant [43].

2.5 Competing technologies

NFC is optimized to transmit small amount of data between very close devices. Furthermore, it permits storage of data in passive devices. In this section we are commenting the main competitors.

Table 6 summarizes the main characteristics of other wireless communication technologies [44]. The main drawback of NFC is its low throughput and the lack of encryption by default. The main advantage is its low power consumption and its quick and easy setup. In general NFC is cheaper than other competitors in terms of power efficiency.

Table 6. Main characteristics of wireless technologies. Sources: Technologies specifications and Wikipedia

	NFC	**Bluetooth (2.1 EDR)**	**Wifi (802.11-g)**	**Zigbee / 6lowpan**	**Wireless USB**
Frequency Band	13.56 MHz	2.4 - 2.5 GHz	2.4 - 2.5 GHz	2.4 GHz 868 MHz (Europe) 900 MHz (America)	3.1 –10.6 GHz
Max Bit rate	424 Kbps	2 - 3 Mbps	54 Mbps	250 Kbps (20 kbps at 868 MHz and 40 kbps at 900 MHz)	53-480 Mbps
Range	Max 10 cm	Max 100 m	Max 140 m	10 - 75 m	3–10 m
Setup time	0.1 s	6 s	?	?	?
Power consumption	15 mA (20mW)	40mA (50mW)	100 - 350 mA (400 mW -1 W)	10 -200 mA (30 -50 mW)	100 mW
Encryption	NFC-SEC (voluntary)	128 bit key	WEP / WPA	128 bit AES	AES-128 with CCM
Chip cost	?	$3 -10$	$10	?	$10

The main technology competitor when using the Reader/Writer mode are the barcodes. Currently there are two types of barcodes namely one-dimensional barcodes (classical barcodes to tag the products) or two dimensional barcodes (e.g. QR code). There are multiple standards for linear barcodes. The amount of data may depend or

not of the size of the code. For example: EAN 128 permits store 48 ASCII characters while Code 128 can contains as many characters as they fit in the paper. The capacity of a QR barcode depends on the version of the barcode and the error correction level. The version of the barcode ranges from 1 to 40. Version 1 contains 21x21 modules (squares). Every version increase adds 4 more squares per side. The error correction is quantified in four levels: L (7% error correction capability), M (15 % error correction capability), Q (25% error correction capability) and H (30% error correction capability). Typically M is the most frequently used. The maximum capacity of a QR code of version 40 and error correction level of M is of 2331 bytes (3391 alphanumeric characters). Figure 17 shows an example of QR and linear barcode.The main advantage of NFC is that a reader does not need to have a line-of-sight. Thus, the tag can be hidden from the user view and the external aesthetic of the product is not affected. A big drawback of QR barcodes is that they need optimal environment lighting. QR barcodes are read using mobile phone camera which do not obtain enough quality images unless the illumination is good. On the other hand, laser-based linear barcodes readers can be used under any lighting conditions. However, this technology is not integrated in modern mobile phones. NFC can be used under any lighting conditions even in darkness. Additionally, the efficiency in terms of data capacity versus size in barcodes is very small. The amount of data that can be stored in a NFC tag depends on the chip and not on its size. Currently, 1K of data is a standard size. Moreover, data in the NFC tag can be protected against unauthorized read/write. It is not possible with barcodes. Furthermore, barcodes are easily vandalized. Painting with a pen or pencil on the barcode or placing a sticker in front of it makes it unusable. Although, NFC can be also easily vandalized (see section 2.3), it requires more elaborate methods. Finally, NFC is based in a global standard while there are multiple standards for linear barcodes or 2D barcodes. On the other hand, barcodes offer several advantages. Firstly, barcodes do not need any specific hardware (phone's camera suffice). Secondly, printing a barcode in a paper is much cheaper than a NFC tag.

Figure 17. The QR barcode on the left contains 84 bytes. It is version 5, with error correction of M EAN-13 barcode. On the right EAN-128 linear barcode.

3 INTERACTION IN INTERACTIVE SPACES

3.1 Overview

The first general-purpose electronic computer, known as the ENIAC (1946), occupied 1800 square meters and weighted 30 tons [45]. In the 80s and 90s PCs arrived to homes. Later, computation moved to laptops and started to be mobile. In the last few years popularity of smartphones and tablets has consolidated the mobility and have initiated the ubiquity of computation where "*processing power so distributed throughout the environment that computer per se effectively disappear*" [3]. Computation is not tied anymore to a single device but it is everywhere. The user interfaces have evolved in parallel with the computation mobility and ubiquity. The ENIAC used drilled cards to define programs. Only highly skillful professionals were able to give instructions to the computer. The text based UIs of the first PCs gave way to WIMP interfaces. This is one of the reasons for the popularization of PCs in late 80's. Users input data in the device using a keyboard and a pointing device while the device provides the output using a screen and speakers. Although WIMP paradigm has been adapted to mobile devices, many aspects of the WIMP paradigm still are very deep in the founding of the UIs for modern devices. Moreover, WIMP is only valid if users are working with a single device. Hence, it is not suitable for services running on Interactive Spaces. There, users interact with environment (objects, walls, doors, surfaces) to provide input to services. A keyboard or a pointing device is not necessary. Furthermore, the output does not need to be provided by a single device but may come from different objects in the environment. Feedback can be anything that we can sense. Interactive Spaces need new affordances, metaphors and interaction modes.

There are multiple classifications and categorizations of user interfaces. Dix et al., for example, list six different classes of user interfaces, namely command language, natural language, menu selection, form filling, direct manipulation and anthropomorphic interfaces [46]. Quigley claims that there exist many types of input technologies that do not fit in the previous six classes since they rely on new devices [47], for example gesture recognition, voice recognition and augmented reality. Furthermore, Quigley suggests new classes, namely Tangible User Interfaces, Surfaces User Interface, Ambient User Interface, Augmented Reality and Multimodal interfaces. These new interface classes are suitable for Interactive Spaces.

In the next sections I will provide an overview of classic interaction methods and the new interaction methods that are suitable for Interactive Spaces. I will pay special attention to Tangible User Interfaces (TUIs) since it is the main interaction method used by REACHeS services. More information related can be found from [47], [48, Ch. 6], [49].

3.2 Classic Interaction methods

The first interface type used in PCs is *Command-Based Interface.* It required that the user gives commands to the machine by typing them with a keyboard, pressing a single key or a combination of these. The response of the system was also provided in text form. A big disadvantage of this type of interface is that the users must remember the names of the commands and their syntax. Nowadays, only some professional users use this kind of interaction.

Command based interfaces evolved to *Graphical User Interfaces* (GUIs) in which commands where represented by icons or shown in menus. A GUI follows the WIMP (Windows, Icons, Menus and Pointing devices) paradigm. The pointing device controls a cursor which selects widgets to interact with. The goal of windows is to provide a way to interact with multiple tasks and applications in the same physical display. Windows can be opened, closed, scrolled, resized and overlapped. Menus offer a hierarchy of commands and options that can be selected using the pointing device. Menus and sub-menus should be designed in such a way that most common commands are easier to find than uncommon ones. There are multiple types of menus: flat lists, flow-down lists, pop-up lists, contextual list and cascade list. It is easier for a user to browse commands in menus than to recall textual commands. However, UI developers must be careful when designing menus for not creating menus with a big amount of elements, especially if the screen size is limited. Finally, the most important aspect of WIMP interfaces for Interactive Spaces design are the icons. Icons represent commands and applications. They are activated by clicking with the pointing device. Icons are used instead of text labels because they are easy to recall and they use the power of visual representation to create metaphors. Sometimes, text labels can be used close to icons to disambiguate their meaning. The mapping between the representation and its corresponding action can be based on similarity (a photo image to represent an image file), analogy (a scissors to represent the command to cut), or common knowledge (for example an X represents a command to close a window). An icon must provide a clear affordance to a user and must be easy to remember. Icons' symbols must adapt to the device in which the GUI is running (e.g. high resolution screens can have more detailed representations). A recent tendency is to make them simple, emphasizing outline forms and using a small amount of colors.

It is important that a GUI adapts to the device limitations. Nowadays, more and more users are performing common tasks like reading emails or browsing the web using their mobile phones instead of the PCs. Porting an application from a PC to a mobile phone involves redesigning the GUI to convey the limitations of the phone (small screen, no pointing device, etc.). Earlier menus were accessed using two way directional keypads or four way navigational pads. In smart phones touch screen have substituted buttons. Portable devices add also new interaction methods that are not possible on PCs such as gesture recognition. Users can interact with applications e.g. by tilting a phone. Application developers must overcome UI limitations by adapting new interaction methods and sensor information, available only for mobile devices, into their applications.

The GUI alone is not enough to provide a good user experience. *Auditory Interfaces* are complementary interfaces that provide information when the eyes are focused in some other task or alert users from system errors. Furthermore, they are used to reduce visual overload, reinforce visual messages and channels emotions. Auditory interfaces offer an extra information channel to deliver information that does not fit on a display. The counterpart of an icon in a GUI is what some researchers call auditory icons: a translation of visual artifacts into auditory artifacts (for example the sound of a paper being wrapped when you empty the trash can of your Windows system). Sound feedback should not be abused. Developers should be careful when using sound feedback in public spaces (privacy and annoyance to other people need to be considered). Auditory interfaces have been used, for example, as an alternative to GUIs for impaired people. They have been widely used in games and lately for ubiquitous location applications as well [50].

The way in which information is presented on a screen determines the way in which a user interacts with an application. One extended way of presenting information is using Multimedia content. Multimedia is a combination of different media (graphics, text, video, sounds and images) within a single interface [51]. Different media content is connected by means of hyperlinks embedded in the media itself. Multimedia is important because it facilitates access to multiple representations of the same information. Each media type is more suitable for certain types of intellectual tasks than others. This leads to a better understanding of the information content. A clear example of the importance of multimedia content is the evolution of web pages.

3.3 Interaction methods for Interactive Spaces

Ubiquitous Computing, Internet of Things, Ambient Intelligence and Interactive Spaces aim to transfer computation from PCs to everyday life objects. This brings new challenges to the way users interact with services and applications. GUIs are not valid when there is no screen, keyboard or pointing devices. The type of interaction available depends on the environment and the context. Some researchers offer as a solution breaking completely the GUI concept and making the interaction more natural [2]. People interact with applications and services in the same way as they would interact with other people or things in the physical world: talking, moving the hands or body, etc. This is what is known as NUI (Natural User Interfaces). There are multiple input methods for NUIs: speaking, manipulating objects and surfaces or making gestures with fingers, hands or the whole body. The next subsections discuss different interaction methods suitable for Interactive Spaces.

3.3.1 Voice, gestural and multimodal UIs

In a *voice user interface (speech recognition)* a person talks to a machine in a natural way. The machine understands what the person says and gives auditory feedback. This feedback is usually a synthesized voice although recording a human voice sounds usually more natural and friendly. The audio signal of a human voice is captured by the system, digitalized and then compared against an acoustic model of speech sounds (formed by dictionaries, grammars and search algorithms). The voice is translated into commands, which are executed by the system. To make the interaction as natural as possible, voice user interfaces should include a feature named “barge-in” which permits interrupting the voice of the machine and giving a new request while the system is still talking. Moreover, the system must provide feedback when recognition was not accomplished. Asking for confirmation, before executing important commands, is also an important design principle. Voice user interfaces have been used widely by the help desks of many companies for call routing: the machine asks users several questions which help the system to decide the right service to handle the user request. It is being used also as a substitution of GUIs in people with disabilities (e.g. blind people who cannot read messages on the screen). Nowadays is being integrated in many ubiquitous systems e.g. to control car navigation systems or to interact with services through the mobile phone (Google Now[18] or Siri[19]).

There are two types of *gestural interfaces*: those in which gestures are made with fingers on a screen and those in which a user makes air-based gestures with any part of

[18] http://www.google.com/landing/now/
[19] http://www.apple.com/ios/siri/

his/her body. I focus on the second category in this section. Non-verbal communication (including body language, tone of voice, etc.) is an integral part of human communication and complements verbal communication. Gestures are examples of non-verbal communication. A gesture is a "s*ymbolic movements that encode meaning in their direction, orientation, and shape. They form an integrated system with speech to which they are semantically and temporally linked [...]. The meaning they encode is closely related to, but not necessarily identical with, that expressed in language and speech [...]*" [52]. Any system aiming to recognize user intentions should be able to detect and understand user's gestures. Gestures are translated into commands that are sent to the system. Nowadays, it is possible to recognize gestures made with any part of the body, for example, face expressions can be analyzed to get information related to the person's mood and feelings. Cameras and other sensors produce data that is processed using pattern recognition techniques. In the case of cameras, a computer vision algorithm processes the video produced by one or more video cameras [53]. The algorithms perform three different tasks: segmentation (find relevant parts in the images such as shapes, skin color, etc.), tracking (follow the movements of certain significant points using for example Kalman filtering) and classification (extract relevant information). Other sensors used to recognize users actions are accelerometers and gyroscopes or infrared sensors. Accelerometers and gyroscopes can be attached directly to person's body or to objects that person is carrying (e.g. a smart phone). Those sensors detect gestures such as shaking, bumping, tapping, bouncing and rotating.

Voice and gesture recognition are useful to create applications that require freedom of movements. Gesture recognition has been used as an innovative interaction mode for the gaming and entertainment industry. The three main games consoles permit interacting with games by gestures: Playstation 3 tracks players' movements using a camera (Eye-toy), Nintendo Wii's recognize gestures made by players' arms using accelerometers while Microsoft XBox 360's Kinect recognizes body gestures using a combination of camera technology, infrared and multiarray-microphones. Designing an intuitive and easy to use gesture interface is not an easy task. Norman claims that majority of gesture interfaces are neither natural or easy to learn [54], [55]. Visibility and feedback are two of the most important design principles on user interaction. It is a big challenge to show the functions and commands available for a system and which gesture triggers each one (visibility challenge). It is also difficult to provide feedback when a gesture made by the user does not trigger any command (feedback challenge). Other critics on gestural user interfaces are cultural issues. Typical gestures such as waving hands or head, pointing or nodding have different meanings in different cultures and geographical areas. These problems are especially important in smart environments where systems try to predict users' intention. However, in Interactive Spaces gestures can be predefined and advertised to the user using any available method.

Humans use simultaneously different input/output interaction modes while communicating with other humans. Multimodal interfaces combine the input coming from multiple interfaces into one. Multimodal interfaces multiply the way information is experienced and controlled by users, increasing the robustness of interaction by using redundant and complementary information (mutual disambiguation). This helps to overcome the limitations coming from a single modality. Usually when we refer to multimodal interfaces we mean a combination of multiple interfaces to accomplish a single task. However, some authors claim that two or more interfaces which

accomplish the same task but are not used simultaneously are also multimodal interfaces. In the last case, users choose among different interaction styles, the one which matches their particular situation. This permits also adapting the interface to multiple platforms or devices each one having its particular look and feel. The most crucial element in a multimodal interface is the definition of the mechanism which fuses the inputs from multiple modalities so a coherent interpretation is achieved. This system must provide a common semantic representation for all inputs, that is, translating different input vocabularies to a common language. Multimodal systems can be classified according to the moment when the fusion is performed. Early-fusion systems have one recognizer which is able to analyze the input from different modalities and to extract common features before sending this data to the integration element. These types of systems are used when the interaction methods are highly coupled (e.g. voice recognition and lips reading). On the other hand, late-fusion systems contain one different recognizer per interaction mode. Data interpretation is performed using the results of the different recognizers sequentially. Implementation of multimodal interfaces poses a bigger challenge than single modality systems. Reeves et al. [56] provides some basic guidelines for designing multimodal systems.

3.3.2 Feedback technologies

Classical WIMP based user interfaces trust in visual and auditory interfaces. Interactive Spaces might not contain neither displays nor speakers or they might not be available at a certain moment (for example other users are utilizing them). In some situations, visual or auditory feedback would provide excessive amount of information to the user. In other cases, context can set up restrictions in the type of feedback that can be used. For instance, it would not be a good idea to provide only auditory feedback in noisy environments. Besides eyes and ears, skin can also be used to provide feedback. This type of feedback is known as *haptic feedback* [57]. Haptic feedback, in general, refers to the sense of touch. Skin can sense surface features as well as tactile perception (tactile feedback). Muscle and tendons can feel another type of haptic feedback known as kinesthetic feedback. This feedback is useful to simulate nature forces (gravitational force, friction, etc.) on virtual objects. Tactile interfaces suit better in Interactive Spaces than kinesthetic ones. Although any kind of stimuli that skin can feel such as pressure, heat or pain are considered tactile feedback, the most common stimuli used is vibration. Vibrotactile interfaces apply vibration to the human body by means of a motor or an array of piezoelectric pins. Typical vibration frequencies range from 10 Hz to 1 KHz. The modification of frequency, amplitude and duration of vibrations produce different effects. Vibrotactile actuators can be embedded either on the user's clothes (wearable tactile interfaces) or any device they are carrying such as mobile phones.

Haptic feedback is used in a big range of applications usually as a companion of other feedback sources (commonly auditory and visual feedback). Current mobile phones use haptic feedback in two different ways: to announce an incoming call when the mobile is in silent mode and to confirm key presses on touch screens. Haptic are nowadays used in gaming console controllers (e.g. steering wheels on driving simulator or Wiimote controller) to enhance user experience. Actuators embedded in clothes are used to give indications to people while they are performing outdoor activities [58] and when they are playing musical instrument [59]. Sometimes, haptic feedback can be used as a unique feedback source providing "*brief signals conveying an object's or event's state, function or content [to create] an expressive haptic*

language for interpersonal communication" [60]. McLean and Enriquez call those signals "haptic icons". A haptic icon is an information entity by itself and does not need of other feedback source to have a clear meaning.

Other important feedback sources for Interactive Spaces are Ambient User Interfaces and Augmented Reality. An Ambient User Interface is embedded in the environment and it is providing continuous output in the form of images, sounds, movements, lights or even smell. That information is perceived by the periphery of user awareness [61]. The system input comes from sensors or directly from a Web service. Augmented Reality is part of a wider research area named Mixed Reality. Augmented reality overlays, in real time graphics, text or sound onto a real-world scene [62].

3.3.3 Surface User Interface

In our everyday life we use tables e.g. to write notes, place books we are reading or show a photo album to friends. We also use walls (either indoor or outdoor) to place posters and advertisements. Traditionally, surfaces have been used to create, show and share non-digital content (e.g. photos in a photo album). A surface user interface (SUI) can be defined as a user interface based on a horizontal, vertical or spherical surface on which images and graphics are projected and in which input methods are integrated in the same physical surface (for example by means of a touch screen). SUI output follows the same principles as classical GUIs design but the big difference is that mice and keyboards are avoided. The size of the interaction area should be taken into account carefully. The size of the icons and distance among them is decided in such a way that common operations like drag-and-drop are not clumsy. Furthermore, rendering of keyboards in this type of interfaces might do the system less ergonomic so it should be avoided as far as possible. SUIs use different technologies to receive the input from the user: touch screen, active stylus and computer vision (gesture/image recognition).

There are at least four types of technologies used for touch sensing namely: resistive, surface acoustic wave, capacitive and infrared. A *resistive touch screen* consists of a glass coated by two thin resistive layers separated by invisible spacers. When a pressure is applied on the glass the two sheets come into contact. A voltage is applied on one sheet in one direction. When there is a contact between two layers the second layer measures the voltage and depending on its value it can calculate the X coordinate. The result of the inverse operation is Y coordinate. Pressure can be generated either with a stylus or with a finger. The big advantage of this technology over the others is the low production cost of these screens. Furthermore these screens are rugged and robust. However, they are slower and less sensitive than the other technologies and the resistive layers do not permit more than 75% transparency. The *capacitive screen* takes the human body capacitance as an input, so they can only be manipulated using bare fingers or special capacitive stylus. The most common capacitive sensors are built using two different technologies: surface capacitance and projected capacitance. Surface capacitance is built by applying a conductive coating to one side of an insulated surface. A small voltage is applied to the layer. When a conductor (e.g. a human finger) contacts the insulated surface a capacitor is dynamically formed. Measuring the change in the capacitance of the layer sensors can estimate the position of the touch. A projected capacitive sensor is built using a matrix of rows and columns of conductive material. A capacitor is applied at each intersection of each row and column. A voltage is applied to the grid. When a conductor

approaches to the panel, it distorts the electromagnetic field modifying its mutual capacitance. The capacitance point at every individual point of the grid can be measured, assuring quite accurate positioning of the finger. *Surface acoustic wave* sensors use ultrasonic waves that pass through the touch screen. When the screen is touched, part of the wave is absorbed. A transducer is able of measuring the power loss of the wave and calculates the position. Finally, *infrared based touch screen* detects the position of objects that interrupt the light passing through a grid of LEDs. Alternatively, a modern technology known as *optical touch* detects the position of any object in a display using two or four sensors placed at the corners of the screen, which detect the interruption of an IR light source placed at the camera position. Additionally, active styluses are pen-like tools that sense the movements of the stylus. An active stylus uses different sensors to recognize the user movements such as cameras, accelerometers and gyroscopes.

Non multitouch screens permit three different gestures: tap, drag and swipe. Tapping is the equivalent to a mouse click and it is used to select elements in the UI. Additionally, the equivalent action of pressing the secondary button is a long tap. Drag is similar to the mouse drag-and-drop gesture and consists on moving the finger on the surface without losing contact. It is used to move widgets and icons from one place to another. On the other hand, flick or swipe consists on quickly brushing the finger on the screen in one direction. Multitouch screens have led to new gestures for interaction: pinch and spread (touch with two fingers and bring them close together or move them apart), and rotations (touch with two fingers and move them circularly clockwise or counterclockwise).

SUIs have made real the Weiser's vision of an ubicomp environment full of "*tabs, pads and boards"* [1]. Smart phones are the tabs, tablets are the pads and finally large screens are the counterpart of boards. Hence, SUIs are a suitable interface for almost any kind of application. For small and medium size SUIs self-illuminated displays are used (LCD, LED and plasma displays). Large scale displays can use in addition front-project displays and rear-projected displays.

Classical WIMP interfaces are designed for a single person. When collaboration is required, usually a user takes the control of the mouse and keyboard, complicating the participation of others. In contrast, large SUIs add an extra dimension to the interaction: the possibility of sharing the interfaces among multiple people enabling collaborative creation of content. Furthermore, users can observe interactions made by others. Designing applications for this kind of shareable interfaces is an interesting research topic, since applications must permit fluid interaction between multiple users simultaneously.

SUIs have existed for a long time (e.g. at ATMs, ticket machines, museum guides, PDA, car control systems). Entertainment and telecommunication industry have popularized this type of interfaces by embedding touch screens in small devices such as tablets, smartphones and portable game consoles. In the case of smartphones, SUIs have completely substituted displays and keypads. The importance of small and medium sized SUIs has increased considerable in the last years. Furthermore, some studies suggest that the SUI is going to stay and the technology is now in the "consolidation phase" of the hype cycle [63]. An application named Reactable [64] introduces and specifically interesting UI since it mixed SUIs with TUIs. Its interactive surface is used as a musical and sound effect instrument designed and implemented by the Pompeu i Fraba University (Spain). Users place cube-shaped blocks with a barcode-like marked on the surface. Each cube represents a

musical-synthesizer tool. Users create different sounds by placing and manipulating (rotating, moving, etc.) the cubes on the surface.

Building Surface User Interfaces using NFC

Near Field communication technology can be used to build restricted but cheap SUIs. The Smart Poster NDEF format defined in section 2.2.5 enhances posters with multimedia content (images, video, audio or webpages) by attaching NFC tags on them. The tags contain hyperlinks to the content itself. When an NFC phone touches a tag, the content is shown on the phone's screen. In this case the SUI is composed by the poster and the mobile phone display. In our research group we have proposed the Interactive Poster (see section 5.4.6) which provides a wider set of services than the ones that can be created using just the Smart Poster NDEF format. Those services can show output not only in the mobile phone but also in external displays. In this case the SUI is a 3-tuple formed by the poster, the external screen and the mobile phone. DOCOMO Euro-Labs researchers have built SUIs using NFC. In the first one [65] the authors use a laptop display augmented with a mesh of 7 x 4 NFC tags. Tags are placed on the back side of the laptop screen. Each tag stores its location in the mesh and the Bluetooth MAC address of the laptop in use. The display shows interactive content, placing icons in the same location as the tags. A user touches the display with the phone. The phone reads the closest NFC tag to the tapped point, connects to the laptop via Bluetooth and submits the coordinates of the tag to the laptop. The laptop changes the content of the screen based on the received coordinates. In the second example, authors scaled this set up to big displays (164 cm x 69.5 cm) [66]. The physical UI is a set of 4 x 6 tiles. Each tile had a grid of 8 x 5 overlapping NFC tags. A projector on the ceiling creates the UI on the tiles. When the user touches the physical UI with the phone, the coordinates of the touched tag are sent to the system which modifies the UI accordingly. The authors tested different interaction techniques such as touch-select, click-select, touch&hold, touch&drop and different gestures. The results shows that short NFC tag reading times is needed to achieve fluid interaction. Current reading speed of around 0.5 seconds is still too slow.

3.3.4 Tangible User Interfaces

GUI presents operable elements (menus, icons and other widgets) on a device's screen. In contrast, TUIs (Tangible User Interfaces) use physical artifacts (e.g. blocks, cubes, balls or toys) as a representation of digital content. Interaction with the system is performed manipulating the physical artifacts (touching, moving, tilting or squeezing). A sensing mechanism can detect a set of actions that a person performs on the object. The actions carried out on the objects are translated into commands. Commands are executed and the system provides adequate feedback. The main goal of TUIs is that users can literally grasp data and commands with their hands. TUIs exploit the affordances of the objects to indicate users how to interact with them. Affordances denote the possibilities for actions that our senses perceive of an object or situation. They are properties of objects that invite us to perform specific actions [11]. A well design TUI should be usable with almost no instructions. User intuition, past experience and try and error should be enough for users to control the system.

Ullmer and Ishii [67] first defined the a TUI as "*[...]a TUI makes information graspable and manipulable with haptic feedback*". However, this definition has evolved over the time [68]. In the initial definition given by Ullmer and Ishii, both the

input and output were provided by physical objects. Nowadays, it is accepted that output can come from any external device located in the environment. An object's physical characteristics provide the afforded metaphors. The position of the object (absolute and relative to other objects), its current characteristics (shape, color, texture, etc.) as well as the action that can be performed establish metaphors to define commands in the digital world. Ideally, users focus on using physical objects (and not computers) and as a result of such interaction a service in the background provides him/her with some information. In recent years the concept of Tangible User Interfaces has been generalized even more with the concept of tangible interaction [69], shifting the design focus to the interaction instead of the visible interface. One of the main advantages of tangible interaction is that tangibles tend to support better collaboration and social interaction [69]. TUIs are used in a wide range of fields [70] such as learning, problem solving and planning, information visualization, tangible programming and music generation. Three main technologies have been traditionally used to create TUIs: RFID, computer vision and a combination of microcontrollers, sensors and actuators.

The concept of "physical browsing" [71] is very tight to TUIs and Interactive Spaces. I consider physical browsing as a particular subset of TUIs where users interact with the surroundings by touching and pointing objects [72]. In this context, touch means bringing a user's mobile phone in contact with the object while pointing is selecting an object by aiming at it with the mobile phone. URLs or other data structures are embedded in objects. When users touch or point at the objects they access to the content of the URLs or interpret the data structure with the mobile device. The main tasks of a phone are: (1) identify the object that has been touched or pointed, (2) read the information stored in the object (if such information exists), (3) process the information or forward it to a system able to perform this task. There are several technologies that enable physical browsing such as NFC, RFID, infrared beacons, laser pointing, barcodes (1D and 2D) and image processing [71].

Tangible interaction in Interactive Spaces using NFC

The previous section claims that RFID and NFC are enabler technologies for physical browsing and tangible interaction. Want et al. were the first who foresaw RFID as a technology to "*build bridges between physical and virtual worlds in a simple way*" [73]. Passive RFID tags are embedded easily in objects and in the environment (e.g. in walls, doors, nameplates or displays). The data stored in a tag permits an RFID reader to identify the action or set of actions to be executed once the tag is touched. NFC brings three main advantages for creating bridges between the physical and the digital worlds:

1. Mobile phone integration. NFC readers are embedded in user's mobile phones. The mobile phones support different communication technologies such as Bluetooth, Wi-Fi, 3G and GPRS which can be used to transmit the data read from an RFID tag to any service in the Internet. Furthermore, mobile phones permit advanced multimodal interaction with users by using: display, keypad, haptic and aural feedback.
2. NFC permits full duplex communication between two NFC devices (peer-to-peer mode). This extends the interaction possibilities between mobile phones and the environment. A mobile phone, for example, can transmit multimedia content to a

NFC enabled display located in the environment when a user brings the phone close to the display. There is no need of another carrier technology, reducing the setup time and saving energy.

3. NFC is standardized. Standards guarantee interoperability between devices from different vendors.

Interactive Spaces do not sense the user actions, but the user directs the interaction with services by performing implicit and predefined actions. Explicit interaction makes the system less prone to errors (provide output only for explicit input) and improves the feeling of the user being in control (something happens only when he/she desires). When using NFC as the main interaction method, the explicit action is to touch an NFC tag or an NFC device. Interaction with NFC tags can be used to (a) start services, (b) control services and (c) select the devices to use in the service.

Before using a service in an Interactive Space, user must be aware of the services that the environment is offering. Icons advertise the services that are available. A service is activated by touching the corresponding icon with a mobile terminal. Other icons might be used to command running services or to select the resources to be utilized. NFC tags are placed behind those icons. An icon advertises a point in the environment that can be touched with a phone (active area). An icon forms, together with the NFC tag located behind the icon, a two-sided interface between the physical and digital worlds. For the user, this interface advertises the action to be executed when the icon is started. For the system this interface contains the information necessary to perform such action. Making an analogy with classical WIMP interfaces the environment is the whole GUI. Users start and command applications by touching icons with their mobile phones; hence icons in the environment have the same role as icons and buttons in traditional GUIs. An NFC tag contains information about the action triggered when the tag is touched. The output is provided either in the users' mobile phone or in some external resources.

The main challenge in icon design is to communicate the icons' affordances. We have studied how to achieve this goal [74], [75], [76]. A user should be able to predict the action to be triggered when the icon is touched with the information given in the icon. If users do not recognize some icons to belong to the physical UI, the corresponding action will be hidden to the user. On the other hand, a user might recognize an icon as a part of the Interactive Space UI, but the meaning of the icon is misinterpreted. Thus, the service activated or the command sent is not the one that the user expects, which impoverishes user experience. Another possible mistake is that the user identifies an existing pictogram wrongly as an icon of belonging to the Active Space. Since touching the icon does not trigger any action, the user believes that the system is broken. To avoid these mistakes, icons are divided in two different areas. The outer part advertises that the icon is part of the Interactive Space UI, while the inner part advertises the action triggered when the icon is touched by means of pictogram, text or a combination of both. All the icons of the UI share the same outer part.

Figure 18 shows a set of icons that have been used to build some of our prototypes. The one on the left shows the general design of the icons. The next icon is used to show a collection of photos in an external display. The third one is used to transfer some information from the NFC tag to the user's mobile device. The last icon triggers a phone call to the number that is stored in the NFC tag.

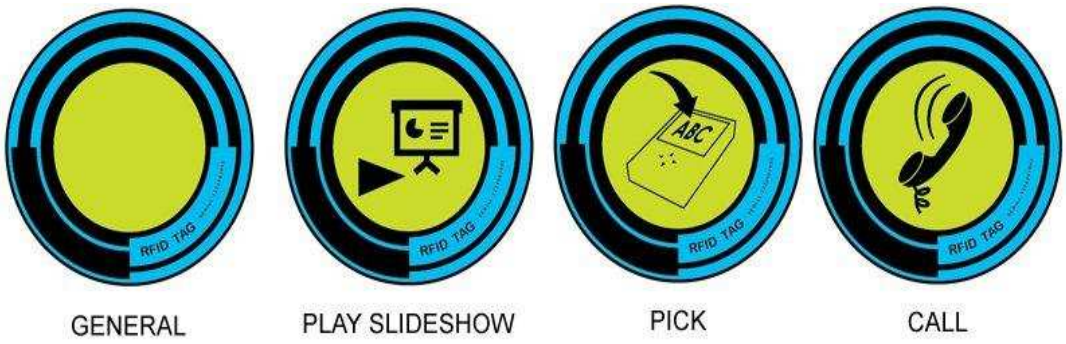


Figure 18. Example icons to identify services and actions in Interactive Spaces.

The pictogram and text shown in the tag might not be enough to fully predict the action to be triggered. For example, which phone number is dialed when the last icon on Figure 18 is touched? What information is retrieved with the icon labeled with "PICK"? Which screen will open a slideshow when the second icon is touched? The extra information needed to answer the previous questions is coded by the position of the tag. For instance, if the last icon in Figure 18 appears on the nameplate of an office, the recipient of the call is the office's owner. On the other hand, if the icon labeled with "PLAY SLIDESHOW" is placed next to a big display, the slideshow is shown in that. Sometimes, the icons cannot be placed directly on the target artifacts. Then, the icon could be placed in a document, poster or sign which represents that entity. For example, in an application that collects multimedia content related to animals living in a zoological park, the icon which enables downloading content cannot be place on the animal for obvious reasons. Thus, it is placed on a plate with information of the animal (name, habitat, etc.) and that is accessible to visitors.

The NFC tag behind the icon contains the necessary information for the system to trigger the right action when the tag is touched. In some situations the tag contains all the information necessary to trigger the application. In other cases, the information is in an external server and the tag contains and identifier or a link (usually a URL) to access such information. Both approaches: data-on-tag and data-on-network [77] can be used in the systems studied in this thesis. The selection of the approach determines the architecture of the whole system. The storage capability of most popular tags types is quite constrained (several kilobytes at maximum) and the transfer rate between the phone and the tag is low. Hence, tags are not a good choice to store multimedia content or big files. On the other hand, reading data from the network increases the complexity and the latency of the system. Hence, we recommend a balanced solution. When the data fits in the tag and it is static data (e.g. a business card), it is preferable to store the data in the tag. When the network connection is not reliable, it is also better to store data in the tag. In other cases, the data should go to the network. As a general recommendation, a tag should contain the necessary data to generate an input to the system. The data stored in the tag should be sufficient, for example, to detect that the tag cannot be used to command the current service.

An NFC enabled mobile phone has a mediator role: it is the tool that the users utilize to interact with the environment and that provides feedback during the process. From the system point of view, a mobile phone is the gateway between the user and the services. A phone reads the data of the tag, preprocesses it and forwards it to the target service. The service response is interpreted by the mobile phone which provides feedback to the user. Using mobile phones as mediators provides the following advantages over other reader types:

- Modern phones have a wide range of transfer technologies including Bluetooth, Wi-Fi, GPRS, 3G or 4G so it is easy to communicate with the system.
- Phones are widely used around the world and hence handy for the users. There is no need for another gadget in user's pocket.
- Mobile phones are personal devices. They store private data, such as personal preferences and user ids. This information can be transmitted to the services when needed.
- Modern mobile phones have rich UIs that can be used to provide feedback after the interaction.
- Modern phones have a wide set of sensors such as accelerometers, gyroscopes, GPS, magnetometer and microphones. This fact facilitates context acquisition and multimodal interaction.

More information on building Interactive Spaces using NFC technology (including icons design) can be found from [75], [78], [79] and [80].

4 REACHES SYSTEM DESCRIPTION

4.1 General Overview

In this thesis I build a software system to study Interactive Spaces. Users should be able to interact with services running on Internet using different interaction modes. The main focus is on tangible interaction and physical browsing using NFC technology. The system I have built in this Master Thesis is REACHeS and acronym of “Remote Enabling And Controlling Heterogeneous Services”.

4.2 The preceding system

REACHeS is the evolution of a previous system that was designed to assist elderly people in their everyday life and to provide them with entertainment for their leisure time. The services are activated by touching RFID tags with mobile devices. Users control services using mobile phones. We initially targeted a simple photo album application as a proof of concept and feasibility study. The photo album service works as follows: several RFID tags are placed on the home of the elderly, located on paper photo albums and photo frames.

The user or a family member stores the photos the elderly wants to watch in a Flickr (www.flickr.com) photo album. When a tag is touched, a display in the room shows the photo album associated to this tag. There are two different versions of the application. In the first one, the photos are shown as a slide show without user intervention (the photo is changed every 10 seconds). In the second version, the user can browse the photos (next, previous, first and last) using the mobile phone keypad as a remote control for the screen. In both cases, the user can close the photo album by closing the application in the mobile phone.

To build this service we implemented a software platform to connect the four main actors of the photo album service, namely: the RFID tags placed in the environment, the mobile devices, the photo album service and the display. The platform provides the following features:

- Services are triggered when a user touches an RFID tag in the local environment. The tag stores all context information needed to start the service. The tag must be visible and easily recognizable.
- A user utilizes a mobile phone keypad to command the service. Mobile phone display must show some general instructions, for example, which event is activated pressing one key. User cannot navigate through phone menus to access different functionalities.
- The system forwards a command from the phone to the correspondent service. It also detects errors (e.g. when there is no target service or no target command) and redirect the response to the mobile phone.
- Services control the content shown on external displays.

.

Figure 19 shows an overview of the system. Figure 20 shows some pictures of the photo album application.

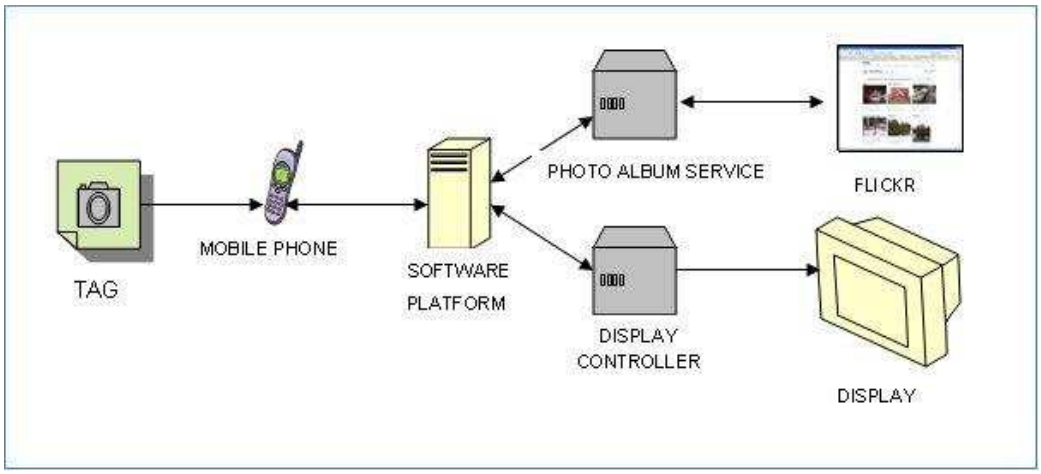


Figure 19. Photo album service.

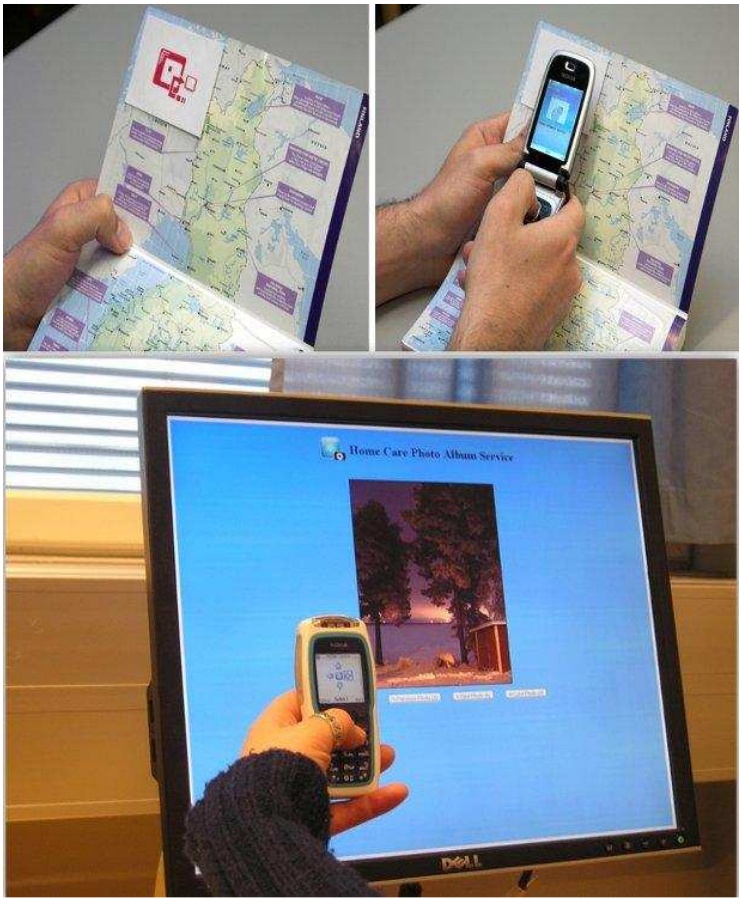


Figure 20. Photo album service. Top: RFID tag behind an icon attached to a book. Touching the icon with an NFC enabled mobile phone starts the application and configures the environment. Bottom: The display showing photos in the photo album. Mobile phone used as remote control.

4.3 System description and main functionalities

The result of implementing and testing the Photo Album laid the foundations of REACHeS, a much more general system, which enables interaction with Interactive Spaces using NFC technology as the main interaction technology. Based on the previous system we identified the following components: User, Mediator, Resources, Services and Interactive Space Gateway (Figure 21):

- User. A user is the person who enters and interacts with the Interactive Space. He or she scans the environment, detects the different services offered and initiates one by touching the corresponding tag using an NFC enabled mobile phone. If the service allows it, the user chooses also the resources to be used. The user selects the interaction method that best support his/her preferences among the ones available for that service.
- Mediator. Mobile phones are not used as traditional I/O devices but as physical objects to interact with the services running in the environment. The NFC tags placed in objects play an important role: firstly they advertise the services to the users and secondly they store the data necessary to start the services. Thus, a mobile phone forms together with the NFC tags a mediator among users, environment and services. A phone has four tasks: receives input from the user through the sensors, transform inputs into commands, send commands to the Interactive Space Gateway and processes service responses giving feedback to the user (audio, images and haptic feedback).
- Resources. Resources are the devices placed in the user surroundings that are used by services to provide output. The most common resources are displays and speakers.

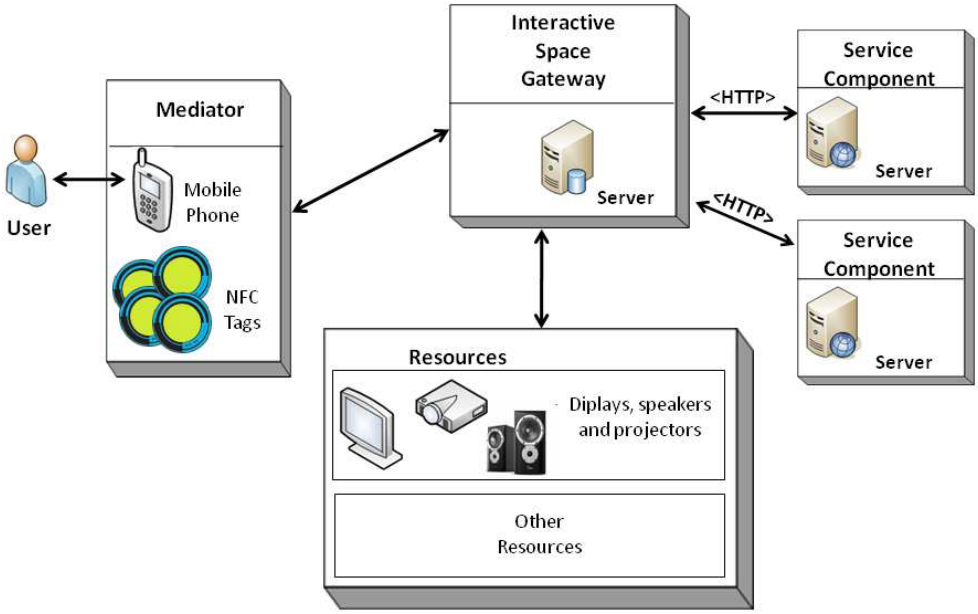


Figure 21. Components of Interactive Spaces augmented with NFC tags.

- Services. Services are a set of related software which provides certain functionality and that can be reused for different purposes. Services receive commands from

users and generate responses to them. They can use resources to provide feedback and show multimedia content.

- Interactive Space Gateway. This component glues all the previous elements together, establishing communication links among them. It forwards the messages received from users' mobile phones to the corresponding service and send back the responses generated by the services. It also connects the available resources to services.

REACHeS system is an implementation of the Interactive Space gateway. The main functionalities of REACHeS are:

- Service registration. Before a service can be used it must be registered in the system by an administrator. During the registration the administrator provides certain information, such as the types of resources that the service requires, the maximum number of clients that can be connected to the service simultaneously and the time that a client can use a service.
- Resource registration. Before a resource can be used by a service it must first be registered into REACHeS by an administrator. The administrator provides some information of the resource in the registration form (e.g. location or IP address) Depending on the type of resource extra information might be needed. For example, displays' information defines its size and resolution. After registering the resource it can be loaded and thus make it available to the services. Current REACHeS implementation supports two types of resources: displays and speakers.
- Resource allocation for services. When a service starts, REACHeS allocates all needed resources for it. Once a resource has been allocated for a service, the service can command the resource at any time. Resource allocation can be done automatically by the system or driven by user commands.
- Interface to control resources by services. Services can control resources through REACHeS.
- Default service to control resources. Resources might decide which service must control it when there are no clients using it. In this case the resource itself acts as REACHeS client.
- Command forwarding. Mobile clients send commands to REACHeS. It processes and forwards them to the target services. Services' response is forwarded to the mobile client.
- Client response adaptation. REACHeS adapts the service's response, so a client can interpret it. (NOTE: This function is only partially implemented in the current REACHeS version)
- Client session control. REACHeS stores session variables that can be created, retrieved and modified by both services and clients. A session starts when a client sends the *start* command to a service and ends when the client sends the *stop* command.
- Error control. REACHeS is able to detect errors in the requests and responses coming from clients and services and to provide error messages to clients.
- Mobile client application supply Over the Air. When a user enters in an Interactive Space, his/her mobile device might not have the clients to control the available services. REACHeS provides a simple way of downloading such applications to the mobile device.

- Timers for services and resources. REACHeS permits limiting the amount of time that a single user interacts with a service or the maximum amount of time that a service locks a resource.
- Content management. Upload and control of files that can be shared by multiple REACHeS services.

We have identified three different actors in REACHeS: the user, the system administrator and the Interactive Space administrator. The user is the person who enters in an Interactive Space and interacts with its services. The administrator is the person who is in charge of installing and maintaining REACHeS infrastructure. Finally, the space administrator is the person who is in charge of setting up the environment. Figure 22 shows the use case diagram for those three actors. The use case scenarios presented in the previous figure are next described in more detail.

User:

- Scan the environment: The user enters in a room and looks around looking for icons advertising services.
- Start a service: The user touches with his/her NFC enabled mobile phone the icon advertising the service that he/she desires to start. In some cases the user might also select resources for that service.
- Command service: The user sends commands to the service using the mobile phone as a remote controller.
- Stop service: When the user wants to end interaction with the service he/she sends the `stop` command to the service. The service goes back to idle state and REACHeS releases all resources associated to this service.

System administrator:

- Register service: When a service is registered, REACHeS receives the information required to access the service (host name, address, path, etc.) as well as the requirements for starting the service (spaces where the service is available, devices required, etc.).
- Unregister service: The service is removed from REACHeS and cannot be accessed anymore. When a mobile client tries to access an unregistered service an error message is sent back. If a service is unregistered while there is an opened session the session is closed and the corresponding error message is sent back to the mobile client. An administrative command permits the system administrator to simultaneously unregister all the services stored in REACHeS.
- Register resource: Resources must be registered in REACHeS before they can be commanded by services. During the registration phase, REACHeS receives basic information about the resource. When a resource is registered, REACHeS assigns a unique id to the resource. Registering a resource is not enough for services to use the resource but the resources must be also loaded in REACHeS beforehand.
- Unregister resource: When a resource is unregistered, it is removed from REACHeS. If a resource is unregistered while still loaded, the resource is automatically unloaded. If the resource is unregistered while some service is using it, REACHeS might reallocate new resources for that service. An administrative command permits the system administrator to simultaneously unregister all the resources stored in REACHeS.

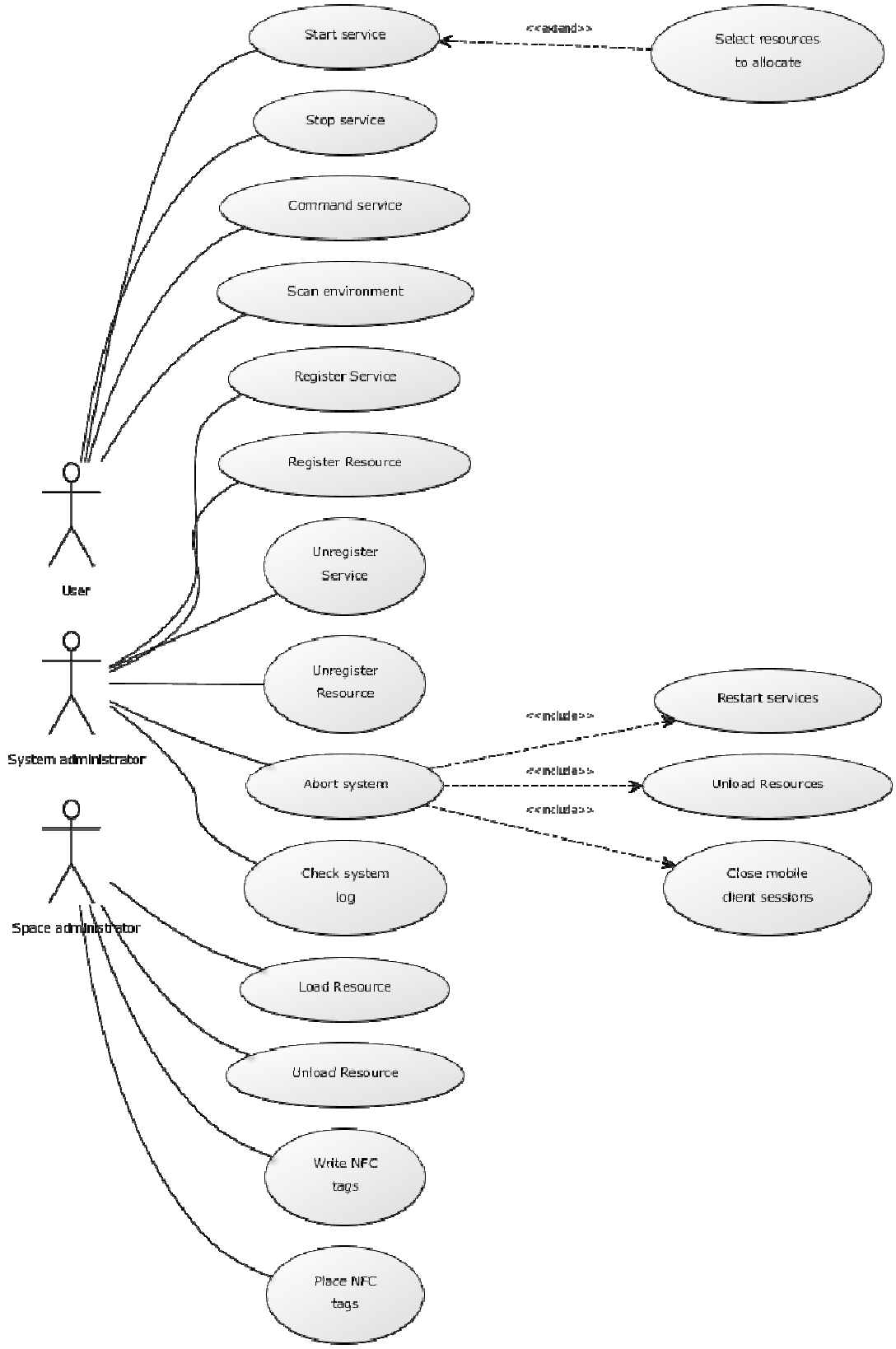


Figure 22. REACHeS Use Case diagram.

● Abort system: When the system is aborted, all sessions with mobile clients are closed, session variables are deleted from the system, services are restarted (all non-persistent data are deleted from memory) and all resources are unloaded. Registered services and resources are kept in REACHeS database.
● Check system log: All commands sent to REACHeS from mobile clients as well as commands sent from services to resources are cached into system log. Logging system can be used to detect problems and bugs in the system and to make statistics of services and resources usage.

Space administrator:
● Load resource: Add the resource to the pool of resources that can be allocated by services. While a service is loaded, REACHeS can always access it, not only to send commands from services but also to commit administrative commands (e.g. get resource status, or identify resource). When a resource is loaded, REACHeS stores an URL that permits access to this resource.
● Unload resource: Remove a resource from the pool of resources available for allocation. When REACHeS receives the command to unload a resource and it is currently in use by one service, REACHeS might allocate dynamically a new resource for that service. If there are no resources that meet the service's requirements, all sessions running on the service are closed.
● Write NFC tags: Writes the content of the NFC tag using an NFC enabled mobile phone. REACHeS provides a mobile phone application to carry out this task.
● Place NFC tags. The space administrator must place NFC tags in the local environment. Tags must be placed behind icons advertising the services to be initiated or the command that is triggered or the resource selected when the phone touches the tag.

An example REACHeS scenario is proposed in Figure 23.

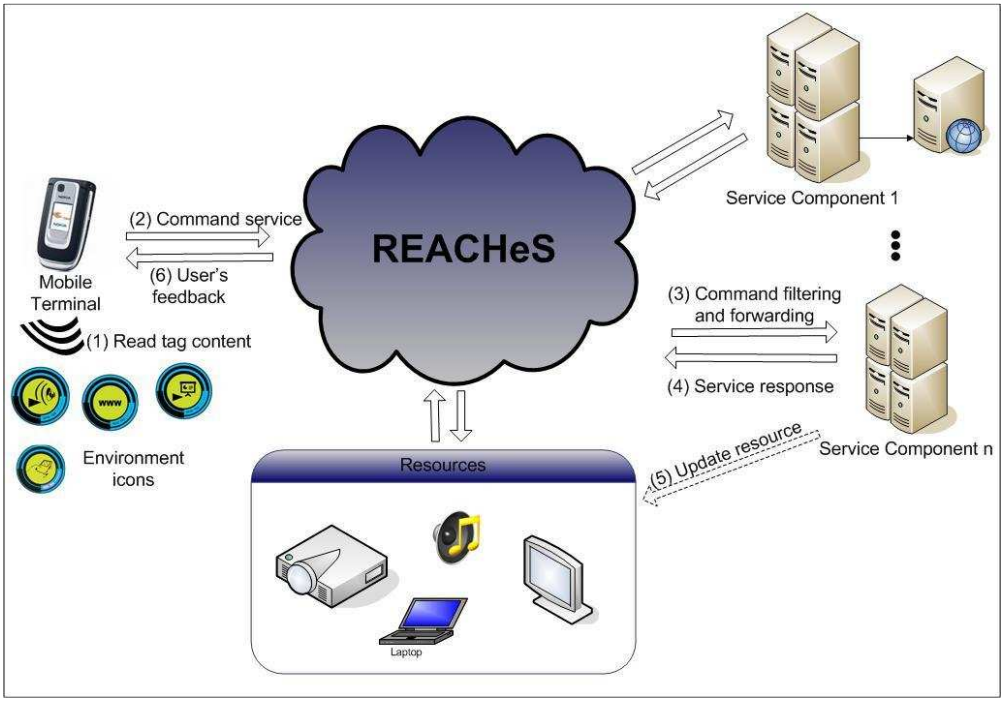


Figure 23. REACHeS system behavior.

In the scenario, a user requests a particular service by touching the corresponding icon with an NFC enabled mobile phone. The phone reads data from the NFC tag attached to the icon (label 1 in Figure 23). The mobile phone starts the client to process the data from the tag. If a client for that service is not installed in the phone it is downloaded over the air. The mobile client processes the data, builds an HTTP request and sends it to REACHeS. The HTTP request might contain some information obtained from the phone configuration files. This first HTTP request contains the '*start*' command to be forwarded to a service (2). REACHeS filters the request and forwards it to the service in question (3). The service processes the request and sends back an HTTP response that determines the feedback to be shown in the mobile phone (4). Simultaneously, the service can also command local resources (e.g. a wall display) using REACHeS as a gateway (5). REACHeS adapts the response send back to the service so it is understood by the phone's client. The response is forwarded to the client (6). As a result, the mobile phone GUI shows a system message or a GUI with instructions on how to control the service. From that moment on, users generate commands by interacting with the environment or with the mobile phone. The processes 2-6 are repeated until the client closes the application.

The sequence diagram in Figure 24 clarifies the behavior of REACHeS. In addition to service control commands (start service, stop service and specific service commands) it includes some administrative commands (register service, register resource, and load resource).

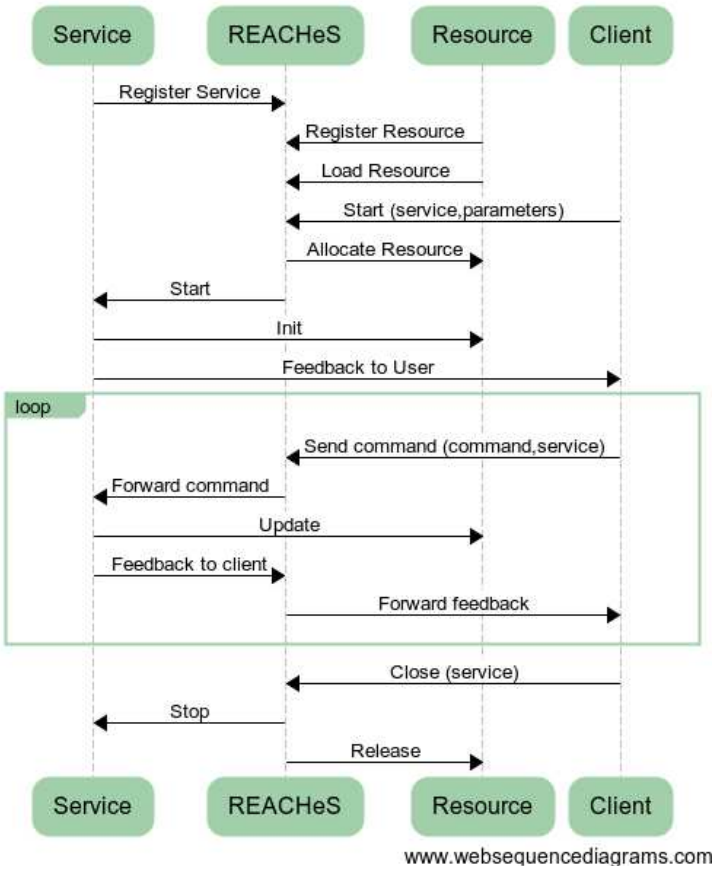


Figure 24. REACHeS sequence diagram.

Communication between the four entities namely Service, REACHeS, Resource and Client is implemented using HTTP protocol. HTTP GET requests sent from the mobile client contain at least two parameters: "`service`" which identifies the target service, and "`command`" which identifies the command to be sent to the service. The request might contain several extra parameters in its URL. Some of them are processed by REACHeS while others are forwarded to services and processed there. When REACHeS receives a message from a client, it filters the parameters, adds new parameters when necessary and forwards the resulting HTTP request to the target service. The service creates an HTTP response and sends it back to the client through REACHeS. The body of the response includes information to produce the adequate feedback in the mobile client. This response is pre-processed first by REACHeS, which modifies the response to adapt it to the client features and limitations. When a service needs a resource, REACHeS performs the allocation using the resources placed in its pool of loaded resources. Services command the resources by means of HTTP POST requests.

REACHeS is transparent to the user and to the services. When a user commands a service he/she perceives that commands are sent directly to the service. Moreover, REACHeS is also hidden from services. They receive requests from clients and send command to resources. Services do not know which resources they are using to show the content.

4.4 Architecture

A REACHeS ecosystem has the following software components: Clients running on mobile phones, the service components running on Internet, resources clients running on resources (displays and speakers) and REACHeS, the server application which connects the rest of the elements together (Figure 25).

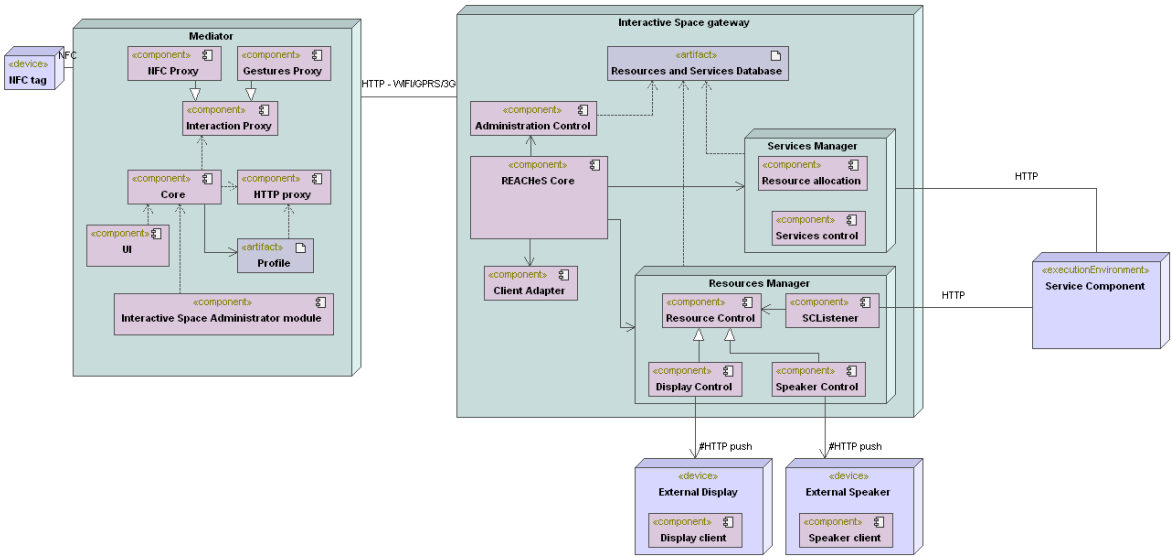


Figure 25. REACHeS Architecture.

Mobile phone clients are native applications running on users' mobile devices. Alternatively, mobile device browsers might be used as mobile phone clients. The application is launched when a user touches a service icon embedded in the

environment. The interaction of the user with the phone produce commands that are encapsulated in HTTP requests and sent to the Interactive Space Gateway. *Interaction proxies* are the components in charge of translating user actions into commands. A client supports multiple interaction methods. It implements a different interaction proxy for each method. All mobile clients implement a NFC *Proxy*. The NFC proxy reads the data on the tag and translates it into a command. The *HTTP proxy* component is in charge of encapsulating the commands originated in the Interaction Proxies to HTTP requests. HTTP proxy might use some personal data or configuration variables stored in a user *Profile* when generating a request. The *UI* module provides feedback to the user by generating sounds, producing vibration patterns or modifying the GUI shown on the mobile phone display. The *core* module orchestrates the mobile client and links the different components together. The mobile client used by the Interactive Space administrator includes an additional module named *Interactive Space Administrator Module.* This module permits setting up the environment to be used with REACHeS: writing and reading data on NFC tags.

REACHeS core is in charge of processing requests coming from mobile clients and forwarding them to the targets. The response generated by a service is translated by the *Client Adapter* module before sending it to a mobile client. This module modifies the response coming from the service to adapt it to the mobile client in question. This component simplifies the GUI in devices with small displays, reduces the size of images in devices with low connections or removes completely the GUI component in the response if the mobile client device does not have a display. Resources and services registered in the system are stored in the *Resources and Services Database.* Other REACHeS components access this database to retrieve the features of the resources and services registered in the system. Additionally, the database holds the pool of loaded resources (resources that are available for the services to use it) and the list of resources that are in use by a service at that moment.

Services Manager module forwards the HTTP requests coming from the mobile clients to the target services. This component is able to send REACHeS administrative commands to the services (e.g. restart a service, inform about network problems or communicate a dynamic modification of the resources in use by that service). Services Manager is also in charge of *resource allocation*. When REACHeS receives a start command it first checks from the services database the type and features of the resources required by the service. Then it tries to perform the allocation using the resources in the pool of loaded resources. REACHeS implements four different allocation mechanisms: fixed, automatic, semiautomatic and manual. REACHeS also supports dynamic resource allocation. If a service is using a resource but for any reason the resource is unloaded, the connection with the resource is lost or the resource timeout is reached, REACHeS tries to allocate a new resource of the same type and features to the service. If this operation is not successful the error is announced both to the service and to the mobile client. The allocation process is explained in more detail in section 4.5.3.

The module in charge of communicating with REACHeS' resources is the *Resources Manager* module. When a resource is added to the poll of loaded resources REACHeS opens a data channel with it. An instance of the *Resource Control* component is in charge of keeping opened the data channel, controlling errors and disconnections. At the same time this component uses this channel to forward commands coming from services or from REACHeS administration components. Each resource type has its own control module implementation. Current REACHeS

version implements a resource control component for displays (*Display control)* and a resource control component for speakers (*Speaker Control)*. The *SCListener* receives HTTP requests from services containing the commands to be executed in the resources. The SCListener processes the HTTP request translating it into a set of commands and send them to the resource. A service does not know the resource in use, but the SCListener must check it from REACHeS database. Finally, the last component in the REACHeS Interactive Space gateway is the *Administration control module.* This module interprets and executes commands coming from the system administrator and Interactive Space administrator clients. Accessing the system log, registering, unregistering, loading or unloading resources are examples of administration commands controlled by this module.

Services are HTTP web services that expose a common API which REACHeS use to send both administrative and client commands. The API is based in overloaded HTTP GET and POST requests.

Resources must implement a resource client which is able to communicate to its corresponding resource control counterpart in REACHeS Resource Manger module. REACHeS is always in control of the data channel although the resource clients are the ones which open the connection when resources are loaded. REACHeS push commands to the resources using any HTTP Push technology. Current REACHeS implementation offers just one-way data channel. Future REACHeS versions might implement a second channel to transmit data from resources to REACHeS.

4.5 Design and implementation alternatives

REACHeS is implemented using Java Servlet technology[20]. Specifically REACHeS is implemented using Java Servlet version 2.5. Tomcat version 5.5 is used as a servlet and JSP container. Tomcat is running on a CentOS server using Java version 6. Display and Speakers control are implemented using JavaScript running on Firefox and Google Chrome. Mobile phone clients are implemented using J2ME technology. Target device is Nokia 6131 NFC, a NFC enabled featured phone implementing the JSR 257 (Contactless Communication API). NFC tags used are Mifare 1K.

Following subchapters explain the design and the implementation of REACHeS main modules. For some modules alternative design options are explained as well.

4.5.1 Core components

REACHeS Core

This is the main component of REACHeS. It is the conductor for the whole REACHeS system. It receives the request from clients (normal users and administrator) and forwards it to the component that can process the request (either a service component or the administration control module). The response received from a service or administration component is filtered using the client adapter and forwarded to the client. REACHeS core is also in charge of keeping the session information. Moreover, it also invokes the resource allocation component before starting or closing a service. If any request is malformed, REACHeS core does not forward it to any other

[20] http://www.oracle.com/technetwork/java/index-jsp-135475.html

component but directly replies with an error message. This module is implemented as a Java Servlet and is using the Façade software pattern.

HTTP Request format. I had three different approaches to implement REACHeS core [81, Ch. 1] once it was decided that REACHeS should be implemented using an HTTP server. First option is the SOAP approach. HTTP POST requests are overloaded. Request parameters including `event` and `service` are inserted into the POST body. I rejected this approach due to the complexity of system and of the HTTP requests and responses. Furthermore, I do not think this is the right approach for REACHeS since the system itself is not providing explicitly a service but it is a gateway that connects different elements of the Interactive Space. The second option is a REST architecture style. I had to reject this option due to the fact that the mobile phone targeted for the client application did not support HTTP PUT and DELETE requests that are essential to implement a RESTful like Web Service. Although there are simple "hackings" in the server side to overcome this deficiency, I opted for the hybrid approach suggested also in [81, Ch. 1] due to its simplicity. This is the most common approach used in commercial web applications.

REACHeS core accepts HTTP GET and POST requests. The service and command names as well as the command and REACHeS control arguments are encoded as URL parameters. Information related to client profiles and session variables are transferred using cookies and other HTTP request headers. The format of the HTTP GET request URL is:

```
http://server:port/reaches/request?service=service_name&event=command_name&command_argument1__name=command_argument1_value&command_argument2__name=command_argument2_value&...&command_argumentN__name=command_argumentN_value&reaches_argument1_name=reaches_argument1_value&reaches_argument2_name=reaches_argument2_value&...&reaches_argumentN_name=reaches_argumentN_value
```

The `service` parameter indicates the service component to forward the command. If `service` is not included in the URL the default service is REACHeS administration service. The `event` parameter contains the command to be forwarded to the service component or to the administration control system. The rest of the request parameters can be divided in two groups: the ones which start with the prefix *reaches_* are parameters that are not transmitted to the service component. Those parameters must be processed by REACHeS and provide information on how to process the request or on how to generate the HTTP response. The parameters that do not contain the *reaches_* suffix are parameters transparent to REACHeS and are forwarded as such to the target service component. For instance:

```
http://server:port/reaches/request?service=multimedia_player&event=start&reaches_resource=00000001&playlist=playlist1.xml
```

This is a URL for a command to start the service "Multimedia Player" which plays videos from the playlist playlist1.xml in the display identified by the id 0000001. The Multimedia Player can be controlled by a MIDlet running on a 6131 phone. Furthermore the User-Agent header is:

```
REACHeS-MIDlet/1.0(Nokia6131;Series40;Profile/MIDP-2.1;Configuration/CLDC-1.1; Resolution/240x320 182ppi)
```

A drawback of this approach is the large size of the URL. Although theoretically there is no limit for the number of characters of an URL some networks and servers limit it. For example, current Apache servers limit the size of the URL to 4000 characters. None of the applications developed using REACHeS up to now have faced this problem.

REACHeS core sequence. Figure 26 summarizes the whole process executed in the REACHeS core. The process starts when REACHeS receives an HTTP request. REACHeS extracts from the URL the `service` and `event` parameters as well as the parameters starting with the *reaches_* prefix and the session id. If there is no event parameter, REACHeS stops the process and sends back an error message to the mobile client. The request is forwarded to the target module. When the response is received, the Client Adapter module modifies the response body and headers and forwards the request to the client. The mobile client might provide some preferences for the Client Adapter as *reaches_* request parameters.

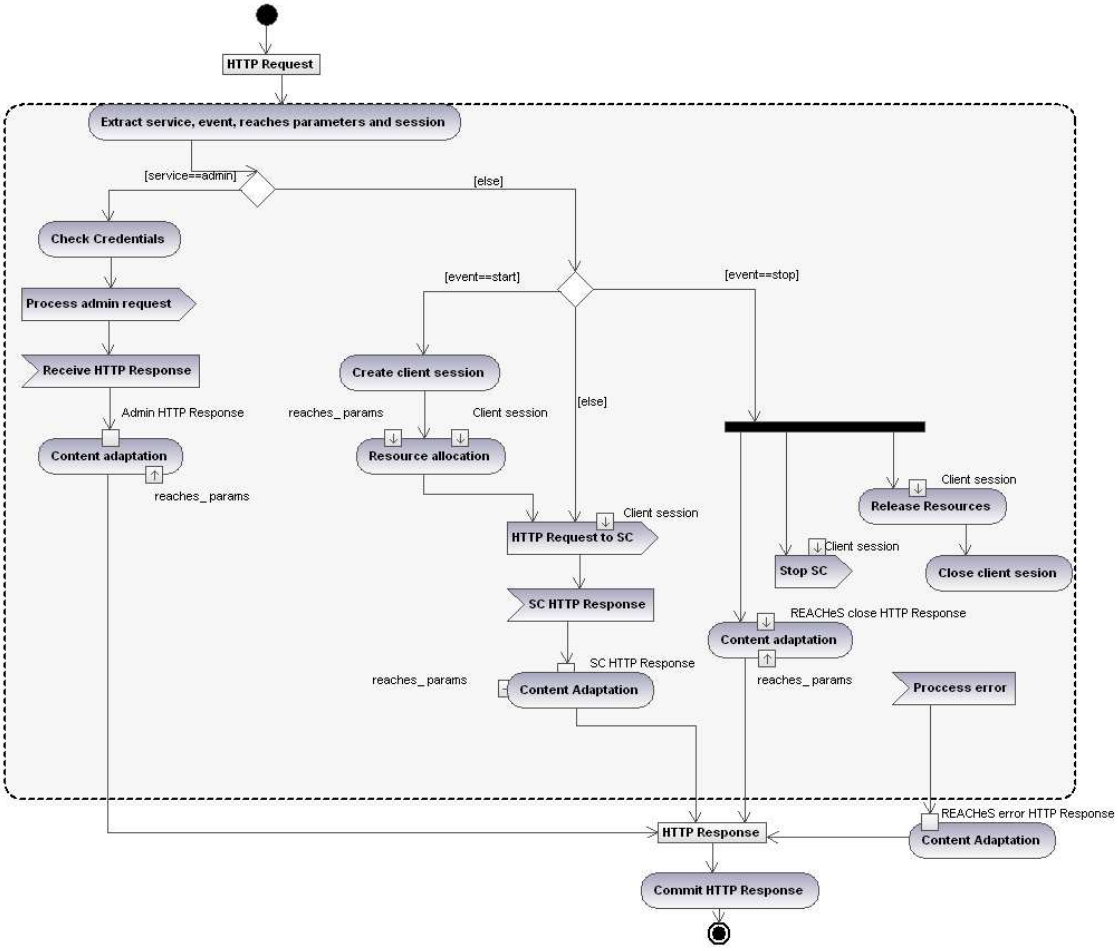


Figure 26. Activity task diagram for REACHeS core.

When the `service` value is “admin” or it is omitted, the command is managed by the Administration Control module. In other cases, the command must be handled by the target service component. If the target service is not registered then an error is sent back to the mobile client.

The commands `start`, `stop` and `restart` are processed by REACHeS before being forwarded to the Service Component (SC). The `start` command initiates a new session (with a unique id and that can store session variables) while the `stop` command closes such session. The session id is shared with the SC and the mobile client using cookies. If the `event` parameter is neither `start` nor `stop` the HTTP request is forwarded to the SC. Once the session is created REACHeS allocates resources before the service starts. This task is executed by the Resource Allocation module. If resources cannot be allocated the service does not start and an error message is sent to the mobile client. After resources are reserved for the service the request is forwarded to the SC. The `stop` command triggers the opposite process: it closes the session and release resources before forwarding the request to the SC. Finally, the `restart` command performs a `stop` and `start`. If REACHeS receives a command other than `start`, and there is no existing session, the command is rejected unless the service accepts "asynchronous commands" and the request contains the parameter `reaches_isAsync=true`.

Error messages. Error messages might be sent back to client because the service doesn't exist, because the remote client tries to send a command before starting a session or because there are no available resources. Error messages can come in 3 different forms:

- Webpage
- Error Status Code: The Status Code indicates why the request generated an error. The Client must interpret this error and inform to the user if necessary.
- Plain text: The message has the format given in the Client Adapter section for a MIDlet.

Administration Control

This module process Administration commands `service=Admin`. This module generates a different response depending on the type of mobile client (see 0). The Table 7 provides the most important REACHeS administrative commands.

Table 7. REACHeS administration commands

Command	Parameters	Description
REGISTER RESOURCE	`resourceFile` or list of service parameters	Stores a resource in REACHeS database and returns a webpage with information of the registered recourse as well as the unique id associated to the resource in an HTTP header (`Resource_ID`). The resource parameters can be provided either In a XML file provided in the `resourceFile` parameter, as request parameters or in a webpage form
UNREGISTER RESOURCE	`id`	Removes a resource from REACHeS database
ABORT	-	Unloads all resources, closes all client sessions and restarts the services

Command	Parameters	Description
REGISTER SERVICE	See REGISTER RESOURCE	Stores a service in REACHeS database. This service is automatically available. The description is similar to REGISTER_RESOURCE.
UNREGISTER SERVICE	`serviceName`	Removes a service from REACHeS database
ABORT SERVICE	`serviceName`	Closes all sessions associated to the service, releases allocated resources and restart the service.
RUNNING SERVICE INFO	`serviceName, format`	Provides information of a running services (service with an open sessions) including allocated resources and clients connected. This command provides information using two possible formats: HTML and XML
RUNNING SERVICES INFO	`format`	Same as RUNNING SERVICE INFO but it returns information of all running services
SERVICES INFO	`format`	Same as RUNNING SERVICE INFO but this command returns information of all services registered in REACHeS
LOADED RESOURCES INFO	`format`	Provides a list of all loaded resources and their properties. Same formats as RUNNING SERVICE INFO
RESOURCES INFO	`format`	Same as LOADED RESOURCE INFO but this provides a list of all resources that are registered in the system
ADMINISTRATION LOG		Returns the administration log file
DEBUG LOG		Returns the administration log file including debug traces
IDENTIFY RESOURCE	`id`	Forces a resource to provide its id to the users

Client Adapter

The Client Adapter receives the `HTTPServletResponse` object coming from the SC and some additional information with the characteristics of the mobile client. The Client Adapter module modifies the body and headers so that the response can be processed in the client. REACHeS classifies mobile clients in six different types:

- **XHTML**: The client is an XHTML-MP enabled web browser which does not accept all response status codes defined by the HTTP 1.1 protocol. Only codes 200

(OK) and 404 (NOT FOUND) are accepted. The HTTP response is an XHTML web page.

- **Handset**: Similar to the XHTML-MP. However, in this case, the web browser accepts all status codes defined by the HTTP 1.1 protocol. The response can be either an XHTML web page or a HTTP status code.
- **Smartphone:** The client is a Handset that can receive any HTML webpage (with JavaScript code). All HTTP status codes are accepted.
- **Computer**: The client is an HTML enabled web browser. All HTTP status codes are accepted.
- **MIDlet:** The client is a MIDlet application running on a mobile device. The response must follow the following format. Note that the "\n " means a line feed. Words in italic are the variables received.

  service=*service*\n event=*event*\n message=*message*\n status=*status*

  Possible values for status are OK or an error message in upper case. Message is a voluntary field.
- **DeafAgent**: The client cannot process the response body but the status code. Only status codes accepted are 200 (OK) 404 (NOT FOUND).

Current REACHeS version performs very basic response adaptation. In future the content adaptation could be improved based on User-Agent data. New groups could be added (e.g. tablet).

The client adapter receives information about the client from the HTTP request:

- In the User-Agent header. The User-Agent string format is currently specified by Section 14.43 of RFC 2616. Basically it is a string containing several tokens informing about the platform details. The system can get extra information from a device using a Device Description Repository e.g. WURFL[21]. This feature is not implemented in current REACHeS version.
- In the *reaches_control* HTTP parameter. The value of this parameter contains the type of client. Possible values are described in the previous paragraph.

4.5.2 Services Components and Services Manager

SCs (Services Components) are services registered in REACHeS. They process requests from mobile clients and other REACHeS services. They must implement the HTTP API defined in Table 8 and Table 9. There are two different types of Service Components: Internal Service Components and External Service Components.

Internal Service Components are important services that are part of REACHeS core and are deployed together with the core system, example services or SC reference implementations. They are tailored to increase their performance. Otherwise they are normal SCs. Since they are deployed with REACHeS they must be implemented using Java Servlet Technology. All internal services extend a common abstract class which provides some helper methods to simplify the implementation of request analysis, response generation and control of resources. These services are not registered using the administration control subsystem, but they are automatically deployed when REACHeS starts. In the future internal services should be installed dynamically.

[21] http://wurfl.sourceforge.net/

External Service Components are deployed in external servers. External services implement two different modules: A HTTP server module which receives HTTP requests from REACHeS' Service Manager and an HTTP client module which commands resources through REACHeS' Resource Manager. The HTTP server module must expose its public API to REACHeS. Developers register these modules dynamically into REACHeS using the Administration Module.

Table 8. Service HTTP request API. Services receive an HTTP GET request with the following parameters.

URL Parameters	
Name	**Value**
`service`	The service to be executed
`event / adminEvent`	The event sent from the mobile client. REACHeS also sends asynchronous administration events to the service (`adminEvent`). Currently only `abort` is supported.
<specific parameter>	All non-reaches parameters sent from the mobile client
Headers	
Name	**Value**
`User-Agent`	One of the six client types shown in section 0
`Cookie:SESSIONID`=<i*d of the session*>;	The id of existing session
Cookie: <*client_cookie*>	All the cookies generated in the mobile phone client are sent to the service without modification

Table 9. Service HTTP Response API. Cookies sent from the services are copied with no modification into the HTTP Response that is sent back to the mobile client.

Status Code	**Header**	**Body**	**Description**
400		Error message String	The service detects an error on the request sent by REACHeS. Error Message describes the error.
204			The service has processed correctly the request. It does not return any value to the mobile client.
204	`responseAddress`		The service has processed correctly the response. The response body is in the URL given by the header `responseAddress`
200		Body	The service has processed correctly the response. The HTTP response body is sent back to the client.

Some services allow multiple clients to share the same session (and hence all clients can control the same resources). In this case, the first client that starts the service creates the session and hence performs the resource allocation. Resources are locked until the last client stops the service. Current REACHeS limits the number of sessions that a service can have to one. Future versions will support multiple simultaneous sessions for services. There is a special type of services, named asynchronous services. They do not create sessions on start. Those services cannot allocate resources and are usually used by other services. One example is REACHeS Mailer service which permits sending e-mails using a predefined SMTP server.

The *Service Manager* is the module which handles communication between REACHeS and the SCs. It is also in charge of resource allocation (see section 4.5.3). It forwards the HTTP request from REACHeS to the SC.

4.5.3 Resource allocation and pairing system

REACHeS performs resource allocation when a mobile client starts a service. The resources required by a service are stored in REACHeS service database. The pairing system assigns a set of available resources to the target service. This information is stored in the service database.

Resources and Services database

REACHeS database is a relational database with four different base relations namely: Registered_resources, loaded_resources, registered_services and running_services. The registered_resources contains a list of resources registered in REACHeS and their characteristics. The loaded_resources contains the pool of resources available for services. Each resource includes the id of the resource in the registestered_resources table and the id of the service that is currently using it. On the other hand, the registered_service stores the properties of all services registered in the system including the URL to access the service. Finally, running_services contains a list of services that are currently running and the session information for each service. Table 10 shows the main attributes of the based relations.

A SQL database would have been the straightforward implementation is. However, this is not the option I chose for REACHeS. The registered_service and registered_resource relations are stored in two XML files. The running_service and the loaded_resource are stored in dynamic data structures. Although this might not be a good choice considering the scalability and performance of the system, I took this decision because firstly, I wanted to deploy REACHeS in any computer with almost no configuration. Setting up a database requires installation and setup. Secondly, the size of the database was thought to be small; just a few services and resources for one single location. Thirdly, the resource allocation was not included in the first REACHeS iteration, so database structures were much simpler.

Table 10. REACHeS database tables and main attributes

BASED RELATIONS		ATTRIBUTES
Registered resources	Common	`id:` Unique id which identifies the resource `resource_type`: display or speaker `defaultController`: service which controls the resource when is in IDLE mode. Can be a webpage `allowsMultipleUsers`: Tells if multiple users can connect to this resource simultaneously `registeredTime`: When resource was registered in REACHeS `lastLoadedTime`: When was the last time the resource was loaded in REACHeS `fixedLocation`: Location of the resource. Cannot be changed during loading. `maxUsageTime`: Max time a service can use a resource
	Display	`bandwidth, quality, security, cpu, resolution,size`
	Speaker	`bandwidth, quality, security, power`
Loaded resources		`resource_id,services,location, ip, port, humanReadableLocation`
Running Service		`name, url, resources,clients`
Registered service		`name`: Unique name which identifies the service `url`: service URL `resourcesRequired`: types of resources that this device needs as well as its characteristics. (includes type of the resource and mandatory and desired properties) `isAsynchronous`: Asynchronous services cannot use resources but do not need a session initialization. `dynamicResourceLoaded`: Can load new resources after session is created `isinternal`: true for internal services `maxExecutionTime`: max time a service can be used `maxNumberClients`: max number of sessions can be opened simultaneously with this service `pairingAlgorithm`: preferred pairing algorithm to make the resource allocation `bandwidth, quality, security,cpu`

Resource allocation methods

REACHeS supports four different resource allocation methods, namely fixed, automatic, semiautomatic and manual. The four methods differ in the autonomy of the system to pick the best resources from the ones in the pool of loaded resources and the

degree of user involvement in the process. The different resource allocation algorithms take into account different factors such as: the characteristics of the available resources, the location of the resources and other context factors, and the requirements for the service. Each service has a predefined allocation method, but mobile clients can override it. In this section I will describe briefly the key aspects of each method. More detailed information related to the algorithms and other characteristics can be found from [82]:

- The **automatic method** aims to start the application while keeping user intervention at minimum. This method assumes that the user does not want to be involved in the resource allocation process at all. The system performs quietly the resource allocation without user intervention.
- The **fixed method** is similar to the automatic method from the user point of view. However, it differs in its implementation. While in the automatic method resource allocation is evaluated dynamically when a service starts, in the fixed method the target resources are either stored in the service registration database or in the NFC tag which initiates the service. Local space administrators associate resources to services while setting up the environment. The environment can be configured in such a way that several tags start the same service but with different resource configurations.
- The **manual method** is an interaction technique which addresses the need of the users to fully control the resource allocation process. The manual method relies on physical interfaces (NFC tags located on or nearby environment resources) to let the user to choose the resources. Users select a device by touching its associated tag. The tag contains the unique id of the resource in REACHeS. The mobile client's interface provides help on how to choose new resources and also identifies some key features of the resources located in the environment. In the resources that cannot be reached easily the tag is placed in a control panel. A user selects the resources after touching the start tag of a service. The start command is not sent to REACHeS until the resources have been selected.
- The **semiautomatic method** shares the resource allocation tasks between the system and the users. When a user touches the start tag of a service, the phone screen shows a list of possible application configurations. Each entry in this list comprises of a set of resources that realizes the service requirements. Each resource is identified using the textual description *humanReadableLocation* parameter from the loaded_resources database. Users can force the resources to identify themselves using the UI in the mobile client. Each resource type identifies itself in a different way. For instance, displays blink for several seconds.

Manual and semiautomatic methods require user intervention after the service start tag is touched. The automatic and semiautomatic methods require internal composition algorithms. REACHeS composition algorithms use genetic and evolutionary algorithms described in [83].

When REACHeS detects that a resource that is being used by a service is not available anymore (e.g. the connection fails) the dynamic resource allocation mechanism tries to reallocate new resources for the service. Services that support this featured are flagged with the `dynamicResourceLoaded` parameter.

Current REACHeS implementation limits to one the number of resources of the same type that can be paired to one service.

Resources allocation and pairing system module implementation

The module is mainly formed by two different subsystems: pairing system and resource allocation. The resource allocation subsystem offers an interface for REACHeS to perform resource allocation before doing the pairing. REACHeS might have multiple resource allocation algorithms. Current REACHeS implementation uses the genetic algorithms presented in [84]. Those algorithms are implemented using C and called by the resource allocation module using JNI. I did not participate in the algorithm generation or implementation. The resource allocation system offers two different points of entry. A Java API to be used internally by REACHeS and a servlet implementation to be used by the mobile clients. The last implementation is used by the mobile client in the manual and semiautomatic method. In this case, the interface with the servlet is an HTTP GET request with the parameter event that can take two values:

- `optimalResources`: used in the semiautomatic method to call the resource allocation algorithm and get a list of resources. The URL contains three extra parameters:
  - `algorithm`: Name of the algorithm to use for the allocation. It must be previously stored in REACHeS
  - `algConfFile`: location of the configuration file for the algorithm
  - `size`: the number of elements in the list
- `serviceRequirements`: used by the manual method to get the list of resources required by a service. For instance: “one display and two speakers”).

Pairing system is a set of utilities classes that performs the resource-service pairing. It receives two parameters. The first one is the service name. The second one is the resource to be paired (for fixed, semiautomatic or manual allocation) or the name of the allocation algorithm (for automatic allocation).

Figure 27 and Figure 28 show a sequence diagram of the interaction between mobile client, REACHeS and the resource allocation and pairing system module for the four different resource allocation methods.

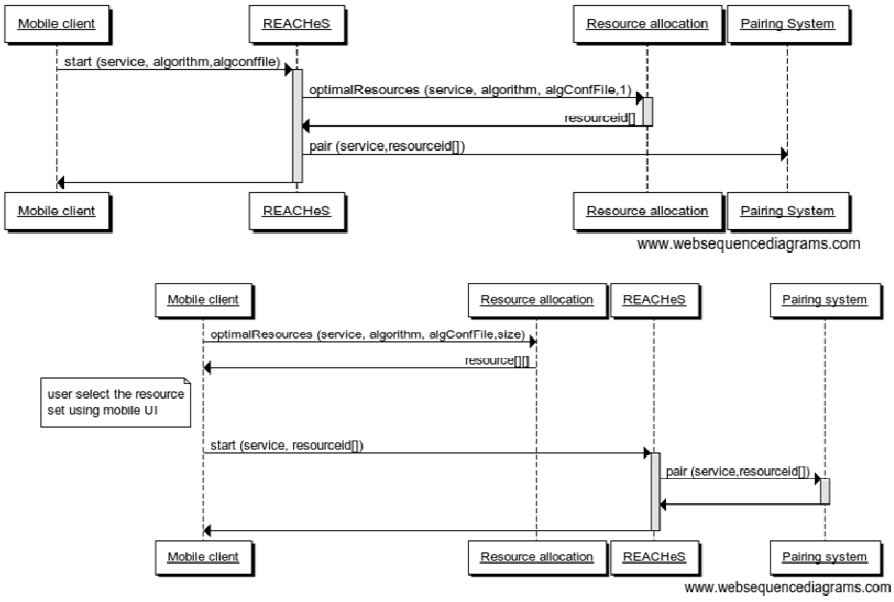


Figure 27. Sequence diagram for semiautomatic (top) and automatic (bottom) resource allocation.

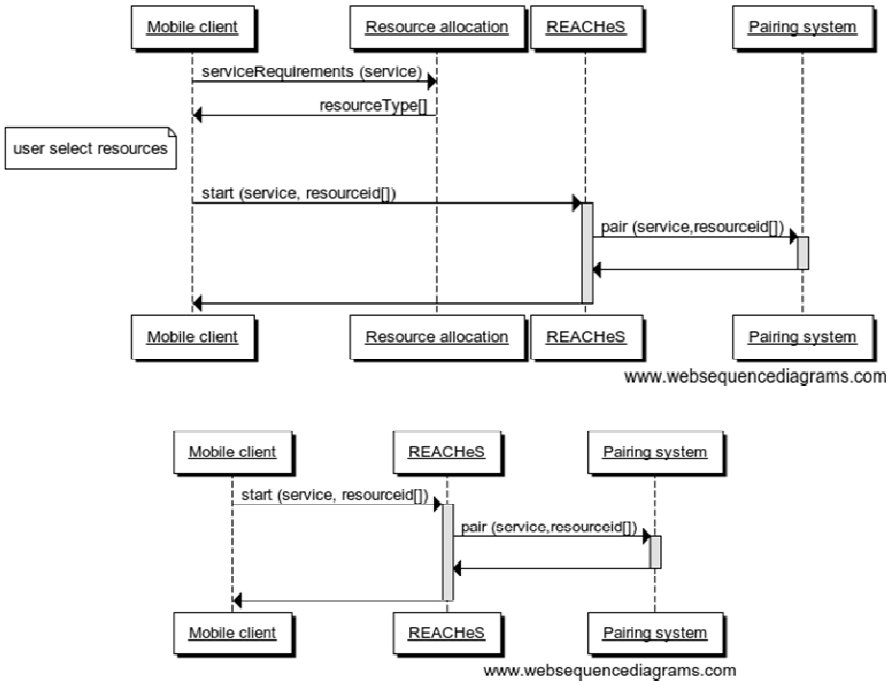


Figure 28. Sequence diagram for fixed (top) and manual (bottom) resource allocation.

4.5.4 Resource Manager

Resource Manager receives commands from services and pushes them to resources. This module monitors also the resources connected to REACHeS. When a resource is not accessible anymore (e.g. network connection fails) it is removed from the pool of available resources for the allocation algorithms. Resource Manager controls two more tasks: dynamic resource allocation (see section 4.5.3) and the "Onstart resource controller". This is a default webpage (for displays), sound (for speakers) or service (for any resource) that automatically runs when the resource is loaded in the system. If the Onstart resource controller is a service, the resource takes a client role while communication to the resource using REACHeS.

From the system point of view, Resources Manager is divided in two subsystems: The "*Service Component Listener"* and the "*Resources controllers*". The first one receives commands from SCs and routes them to target resources. It accesses REACHeS database to learn the resources paired to a service. The connection between REACHeS and a resource is handled by a Resource Controller instance. It keeps a communication channel open to the resource and sends commands through this channel. In the resource side a client process the messages coming from REACHeS. The Resource Manager packages the commands using a protocol that the resource understands. Hence, the Resource Controller is a gateway between REACHeS and the resource.

Service Component Listener

The Service Component Listener offers two different interfaces, one for the Internal Service Components and other for the External Service Components. The first case uses an internal interface implemented in the `InternalSC` (abstract class for all internal

SCs). This module offers an HTTP interface which accepts HTTP POST requests for external services (Table 11).

Table 11. HTTP POST request format accepted by the Service Component Listener

HEADERS	
`service`	Name of the service which makes the request
BODY	
List of events to send to the display. Each event has the following format: <event target="*resource_type*">*event_name paremeter1 parameter2 … parameter n* </event>/n/r Where: • target: Type of resource that should receive the command (e.g. display, speaker …) • event_name: The name of the command to be sent • parameters: A list of parameters separated by white spaces. Note: If a parameter contains whitespaces itself, it must be surrounded by "" (quotation marks).	

The Service Component Listener determines the target resource using the `service` header and the `event` element's `target` attribute contained in the HTTP POST request. Then, it forwards the list of commands to the calculated resource through its Resource Controller instance. The whole process is summarized in Figure 29.

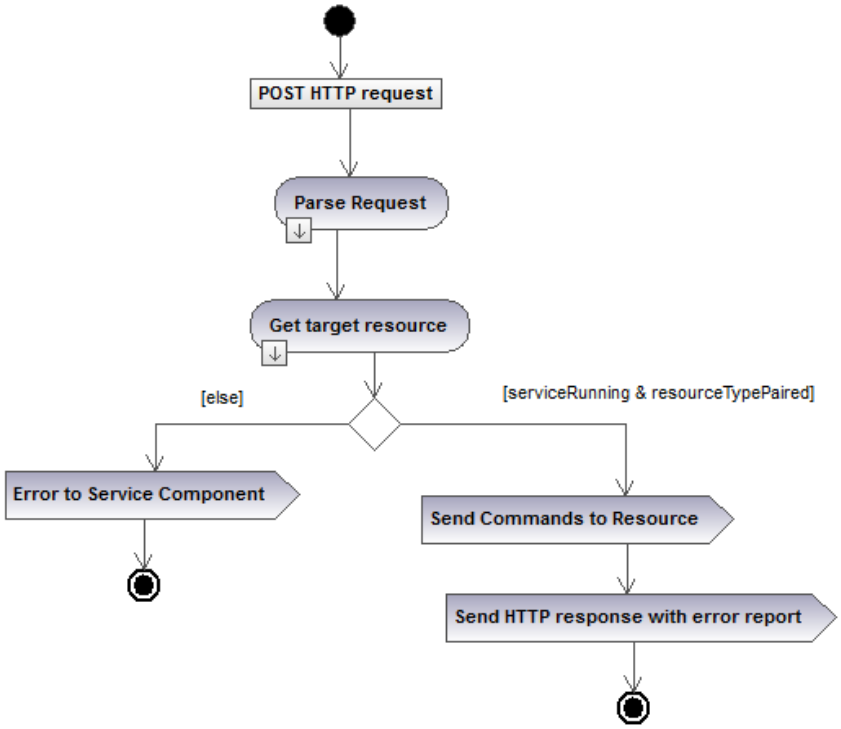


Figure 29. Flow diagram of the SCListener behavior.

Current SCListener implementation has the following drawbacks. First, there is no service authentication. Any HTTP client is able to command any resource controlled by a service by tampering the service id. A combination of digital signature (e.g. providing a unique key when a service is registered in the system) and session id might avoid this possible attack. Second, current REACHeS implementation accepts just one instance of Resource Controller for each Service Component. Hence, a service cannot control two resources of the same type. The main reason behind this decision is to simplify the API between services and REACHeS and the services implementation. To support multiple resources, a service must determine the target resource in the request. Hence, services must track the resources assigned to them.

Resources Controllers

Each Resource Controller instance controls the communication between REACHeS and a resource loaded in the system. Each type of resource (display or speaker) defines its own communication protocol. Even resources of the same type could define different communication protocols. Resource Controller is an abstract class which defines the interface with REACHeS. Each resource implements its own version of the Resource Controller realizing the protocol between REACHeS and the resource. Table 12 defines the methods that must be implemented by the Resource Controller class. Commands are sent to the resources through the communication channels opened by the Resource Controller. The resource may receive two types of commands: administrative commands (e.g. identify, error) and service components' commands.

Table 12. Resource Controller Interface

METHOD	DESCRIPTION
`static createInstance(String resourcid)`	Creates a new instance of the resource controller for a given resource.
`sendEvent (String name, String [] parameters)`	Send an event to a loaded resource
`load()`	Creates a connection between the resource and REACHeS
`restart()`	`unload ()` and `load()` again the resource
`reset()`	Puts the resource in its inital state
`unload()`	Close the connection with the resource
`notifyError(String error)`	Send an error message to a resource
`identify()`	Asks the resource to identify itself. A display could show a symbol on it or a speaker could play a buzz sound.
`destroy()`	Release system resources occupied by this resource. Usually called before `unload()`.

Current REACHeS implementation supports two types of resources: displays and speakers. Displays and speakers are connected to a laptop which implements a JavaScript resource client for REACHeS. The UI is built using HTML5. Furthermore, the both resource clients provide support to control remotely an HTML5 multimedia player (see www.jeroenwijering.com).

The Resource Controller must send asynchronous events to the Javascript client. Hence it is necessary a technology to "push "content from REACHeS to the resource client. I have researched and evaluated the following technologies, all of them, except X11, used in the Web:

- X11: Network protocol for remote GUI and rich input device capabilities. It is used to run client applications on personal computers on Unix-like systems. It provides the basic framework, or primitives, to build GUI environments. Implementation a system based on X11 on REACHeS is too complex.
- HTML refresh: Old technique based on the `<META HTTP-EQUIV="Refresh">` HTML tag. This tag forces the browser to refresh the webpage shown on the display after a timeout. This approach does not permit controlling dynamic webpages.
- Flash: Flash supports sockets using the XMLSocket object. The server side could use one of those sockets to push data to the server. Socket connection uses TCP or UDP protocols to transmit data. Flash runs inside browser plugins.
- Java Applet technology: Like in Flash, java applets support socket connections. Java Applet technology runs as plugin in a browser.
- Web Sockets[22]: A a technology developed by the HTML5 initiative. It is partially supported by current browsers. It defines a full-duplex single socket connection over which data can be sent asynchronously between a server and a client. The advantage of this technology its native support by the browsers themselves. Furthermore, it is built over HTTP so it permits to traverse firewalls and proxies.
- Pushlets[23]: Publish/subscribe framework, with AJAX support and in which the server side is built using Java Servlet technology. It uses persistent HTTP connections: The client makes an HTTP request to the server using AJAX. The server creates an HTTP response object but keeps the `OutputStream` opened. The output stream is used to send data to the client. The client processes this data asynchronously by polling the AJAX response object.
- Long polling: This is similar to pushlets, but the HTTP connection is not constantly opened. The client makes a request and if the server does not have the information ready yet it keeps the connection opened until the information is available. Then, the server sends the HTTP response and closes the connection.
    - Comet and CometD [85] : Also known as reverse AJAX. Comet uses long polling to push messages to the client. Protocols run over HTTP and WebSockets. CometD is the evolution of Comet which uses the publish/subscribe paradigm and the Bayeaux communication protocol. This protocol abstracts the concept of channels and sessions and permits sending data and control signals through the same HTTP connection. It uses chunked transfer encoding to send several responses through the same channel.

[22] http://www.websocket.org/

[23] http://www.pushlets.com/

- BOSH[24]: Emulates TCP transport primitives through HTTP. It has been widely used as transport protocol for XMPP. It is claimed to be more efficient and with less latency than Comet. It uses XML as an envelope for data and control signal. It uses multiple HTTP connections to allow bidirectional communication. When there is no traffic in any direction for some time, the server closes the connection and the client triggers a new one. This is a substitute of "ping" signals in a TCP connection to detect broken pipes.

Table 13 and Table 14 present the commands supported by the display and speaker resources.

Table 13. Common display and speaker events

Command	Parameters	Description
`playAudio`	URL	Plays the audio file which URL is provided as parameter
`stopAudio`		Stop the audio file that is being played.

Table 14. Display events

Command	Parameters	Description
`insertBody`	body	Render the HTML code found in the body parameter
`insertPage`	url	Render the webpage found in the url parameter
`changeAttribute`	id, name, value	Change the value of the HTML attribute "name" contained in the element with id=id.
`changeElement`	id,value	Replaces the content of the HTML element which has the same id attribute as the one provided as parameter. New content is given in the value parameter.
`insertPlayer`	id, destination, filelist, width, height, extraArguments	Insert a multimedia player in the HTML element which contains an attribute id that coincides with the destination parameter. Filelist contains the URL to a document containing the files to be played.
`sendEventToPlayer`	id, event, arguments	Send event to the multimedia player which id=id. Extra arguments can be passed, if the event needs it. The default built-in player accepts the following commands: start, stop, next, previous, volume up, volume down, forward, backward.
`getNextLink`		Move the focus to the next hyperlink on the webpage
`click`		Click on the link that has the focus
`scroll`	up | down	Move the screen scroll
`runJS`	JSON object	Evaluate the JSON object given as parameter and execute the function named "*main*".
`showSystemMessage`	message, time	Displays a message box in the screen

[24] http://xmpp.org/about-xmpp/technology-overview/bosh/

4.5.5 REACHeS Client

REACHeS clients are composed by an application running on an NFC enabled mobile phone and by NFC tags embedded in the environment. The mobile phone is the mediator between the user and the environment. The client receives commands from the user, transforms them in a request that REACHeS understand, processes the corresponding response, and provides visual and auditory feedback to the user. User commands the services using different interaction modes e.g. touching NFC tags distributed in the user environment or waving the phone following certain patterns. Current REACHeS client supports three different interaction methods: NFC, gestures and phone's keypad.

Client implementation

The technology chosen for client implementation is biased by the NFC enabled mobile phones available when REACHeS was in developing phase. The Nokia 6131 (Figure 30) was the first option available. It is a shell-like phone, with traditional 12 buttons keypad and a small non-touchable screen of 240 x 320 pixels. Although the phone is out of the market, it is still today one of the best options to interact with Interactive Spaces. The NFC reader is behind the screen at the top size of the phone. The inclination existing between the screen and the keypad makes it easy to touch tags in vertical surfaces. Other NFC enabled phones are rigid and have the NFC reader in the back side of the phone. Thus, touching tags in vertical surfaces is quite inconvenient. However, they provide better user experience on horizontal surfaces.

Figure 30. Nokia 6131 NFC phone.

Nokia 6131 is a Nokia S40 3rd edition phone, hence applications are developed using J2ME (Java Mobile edition) technology. REACHeS clients are MIDlet applications installed in the mobile device. The main J2ME APIS used by the client application are CLDC (JSR 139), MIDP2.0 (JSR-118), the Contactless Communication API (JSR-257) for NFC communication, the Mobile Media API (JSR135) for playing sounds and the Bluetooth API (JSR-82).

The gesture recognition interface uses a library implemented by Mikko Kauppila available at http://www.oulu.fi/cse/download. It is an accelerometer-enabled gesture recognizer based on Hidden Markov Models. REACHeS classify this client in the MIDlet group (see 0). Hence the HTTP response body is of the format:

`service=`*`service`*`\n event=`*`event`*`\n message=`*`message`*`\n status=`*`status`*`.`

Each service implements its own client (although several services can share the same client). I have implemented my own REACHeS framework library (MCJ) to help with all REACHeS related tasks (e.g. read data from NFC tags, create HTTP requests, process the HTTP responses or communicate via Bluetooth to nearby computers). The framework defines all the components presented in Figure 25 for the mediator except for the Core and the UI. It offers a lot of helper methods for the UI, though. In general, the architecture for different clients is quite similar. The main variations are in the GUI and in the feedback given to the user (auditory, visual and haptic) after processing REACHeS responses.

Alternatively, phone's browser can be also a REACHeS client. A browser permits building REACHeS client also in desktop computers or in mobile phones without the need of installing a specific client application for each service. In the first case, the REACHeS Content Adapter classifies those devices as "Computer" while in the second case devices are classified in the "Handset" (advanced phones) or "XHTML" (older phones) groups (see section 4.5.1). In this case the client GUI is a webpage generated in the Service Component. The page contains a set of links (<a> HTML tags), each one linking to a command that can be sent to the service. The advantage of using this approach is that there is no need to install any extra application on the mobile phone. This approach permits testing REACHeS with other phone models not supporting NFC or even with a desktop computer or a laptop. The disadvantage is that no other interaction mode other than the keypad or touch screen can be used. Gesture control and NFC interaction needs native libraries that cannot be used in the browser.

Communication technologies

Communication between the mobile client and REACHeS uses the HTTP protocol. Any data bearer technology is valid. Since 6131 does not support Wi-Fi the initial approach was to use GPRS technology. However, this technology turned out to be really slow and expensive so I opted to use Bluetooth technology. REACHeS implements a Bluetooth gateway that can be installed in any computer. The gateway receives data via Bluetooth from the mobile client, transforms this data in an HTTP request, sends the request to REACHeS, receives the HTTP response and forwards it to the client using Bluetooth.

Bluetooth gateway is implemented using J2SE (Java) version 6. It uses Bluecove[25] library (a JSR-82 implementation) to access Bluetooth functionality. The communication between the gateway and REACHeS is implemented using Jakarta Commons HTTPClient[26]. The Bluetooth Gateway exposes a Bluetooth server using the Bluetooth Serial Profile (BTSPP). The HTTP requests from the phone are encapsulated into an XML file and sent through the Bluetooth serial port to the gateway. The gateway transforms the XML file into an HTTP request that is sent to REACHeS using either Wi-Fi or Ethernet. The HTTP response is processed by the Bluetooth gateway and encapsulated into an XML file which is sent to the REACHeS client via Bluetooth. Bluetooth client library is included in the MCJ framework. An XML serialized HTTP response in Figure 31 while the request is shown in Figure 32.

[25] http://bluecove.org/

[26] http://hc.apache.org/httpclient-3.x/

```
<?xml version="1.0" encoding="UTF-8">
<response>
    <headers>
        <header>
            <headerName>Content-Type</headerName>
            <headerValue>text/plain</headerValue>
        </header>
        <header>
            <headerName>Accept</headerName>
            <headerValue>text/plain</headerValue>
        </header>
        ...
    </headers>
    <body responseCode="200"><![CDATA[event=start
                                    service=multimedia_player
                                    status=OK]]>
                                    </body>
</response>
```

Figure 31. REACHeS HTTP response serialized in XML.

```
<?xml version="1.0" encoding="UTF-8">
<request>
    <headers>
        <header>
            <headerName>User-Agent</headerName>
            <headerValue>REACHeS-MIDlet/1.0 (Nokia6131;Series40;
Profile/MIDP-2.1 Configuration/CLDC-1.1; Resolution/240x320
182ppi</headerValue>
        </header>
        <header>
            <headerName>Accept</headerName>
            <headerValue>text/plain</headerValue>
        </header>
        ...
    </headers>
    <url
method="GET">http://server:port/REACHeS/request?service=multimedia_pla
yer&event=start</url>
    <body></body>
</request>
```

Figure 32. REACHeS HTTP request serialized in XML.

Data in NFC tags

NFC tags distributed in the environment are a component of the client UI. A tag contains the necessary information to start the adequate MIDlet in the mobile device (if it is not yet started) and the parameters to be sent in the corresponding REACHeS request. A REACHeS tag stores a single NDEF Message that has three different formats namely client-specific NDEF message, general NDEF message and resourceId NDEF message. NFC tags containing client-specific messages are those which are linked to a specific software application in the device. They cannot be recognized by other clients. The tags used to start a REACHeS service belong to this type. On the other hand, tags containing general messages are understood by any

REACHeS mobile client. Finally, an NFC tag containing a `resourceId` identifies a resource in the environment.

A general NDEF message (see section 2.2.5) contains several NDEF record with type = `urn:nfc:ext:isg.ee.oulu.fi:parameter`. The payload is an extension of the Text NDEF Record type (section 2.2.5) with the following parameters:

- lang = "`en`"
- encoding = "`UTF-8`"
- text ="`parameterName = parmaterValue`" where `parameterName` and `parameterValue` are variables.

Each NDEF record is mapped to an URL request parameter in the REACHeS HTTP request.

Specific Application NDEF record	URL	General Command NDEF Message

Figure 33. Structure of an application specific NDEF message.

The client specific command structure is shown in the Figure 33. The specific Application NDEF record is an empty record with an NDEF external type. The type is recognized only by one application in the mobile device. The application is launched after touching a tag. The URL is an optional NDEF Record. It stores the URL to download the application to handle the content of this tag. Finally, the General Command NDEF Message is defined in the previous paragraph.

The resourceId is a General Command NDEF Message with two NDEF records of type = `urn:nfc:ext:isg.ee.oulu.fi:parameter.` The first one defines the type of the resource that has been touched (display, speaker…) and the second one is the id of this resource in REACHeS.

5 REACHES INTERACTION MODES AND APPLICATIONS

In this section I describe how users can interact with REACHeS and hence with the environment using the mobile phone as a mediator. REACHeS supports three interaction modes in its current version: NFC (touch-environment interaction), gesture recognition and touch screen and keypad interaction (classical interaction). In all the cases the functionality of the clients is similar. A user action (tapping a button in the phone screen or touching and NFC tag) triggers a command that is forwarded to REACHeS. The mobile phone generates the GUI according to the service's responses. The phone is merely an interface between the user and the service. The main service content is shown on external screens at the environment. The mobile phone offers the UI to command services and provides basic feedback, for example, it informs the user when an event is not recognized by a service. That information is either shown in the phone display or provided by haptic or auditory feedback. In this thesis I have not studied multimodal interaction. That is, a user cannot simultaneously use different interaction methods to command services. When the REACHeS client starts in the user's device it is bound to one interaction mode.

5.1 Multimedia player: an example application

REACHeS Multimedia Player is an example application that is presented here to show the different interaction modes available in REACHeS. The Multimedia Player service permits a user to control remotely a video player shown in a wall display loaded in REACHeS. When the mobile client starts, it sends a start command to REACHeS which performs resource allocation. Then, the service loads the content of the playlist that is specified in the start command. The playlist determines the URL of the files that are played (they can be stored locally or in Internet). A user can play, pause, restart and stop a video. He/she can also move to the next and previous file in the playlist (Figure 34).

Figure 34. Multimedia player application shown in a wall display. On the left the phone GUI for the keypad interaction on the right a control board for the NFC interaction

5.2 Non NFC interaction modes

5.2.1 Touch Screen and Keypad

In this case the user interacts with the service using conventional WIMP interaction: the phone keypad or the touch screen. The mobile client is started generally by touching a tag in the environment, although a user can also start the application through the phone's menu. In the latter case, the necessary data for resource allocation such as target resource or resources requirements must be provided either by the phone (e.g. it is stored in phone's memory) or by the user. User sends commands to REACHeS by pressing a key or a combination of keys in the keypad (mainly arrows and soft keys) or tapping a widget (in the case of touch screens). The MCJ REACHeS framework for S40 supports this kind of interaction. Figure 35 shows an example UI implemented for the Nokia 6131. The buttons on the GUI are bound to the commands that can be sent to the service. The same interface can be implemented easily as a webpage.

Figure 35. The screen on the right shows the initial state of the Multimedia player application. The UI on the left shows that play has been selected. User browse the buttons using the arrow keys and select a command using the action button. When the action button is pressed the selected command is sent to REACHeS.

One of the main disadvantages of this interaction mode is that users have two different focuses of attention: the phone screen and the external resource where the application is shown. This can be partially overcome if the UI shown in the phone is mirrored in the external screen and every action in the phone is automatically replicated in the external screen. In such a case, users can interact with the application looking only at the external screen and using the phone's keypad to command the service.

5.2.2 Gesture sensor

In this case the user interacts with the service performing gestures in the air with the arm that holds the phone. The gestures are sensed by the phone's accelerometer and gyroscope and recognized by an algorithm based on hidden Markov models described by [86]. The mobile client has a one-to-one mapping between gestures and commands. So, when the user performs a gesture in the air, and the gesture has been defined for the application, the mapped command is triggered and the corresponding HTTP

request is sent to REACHeS. The mobile phone provides haptic feedback (vibration) when a gesture has been successfully detected. Table 15 shows the gestures used in the Multimedia player application.

The main disadvantage is that the users must know by heart the mapping from gestures and commands. Furthermore, some of the gestures might not be natural for the user. One possible solution is to let the users create the mapping themselves but this does not solve the recalling problem. Wall displays might advertise the commands and its associated gestures for the application running. The phone's client could include help screens with the same information. Moreover, this method works to generate simple commands, but it is rather difficult to create commands that need some parameters. A command and its possible parameters values must have a unique gesture (or a combination of gestures). The gesture set might grow to a size that is not usable anymore.

Table 15. Gestures to control the Multimedia player.

GESTURE	ACTION	DESCRIPTION
Punch forward	Play / Pause	Plays the current video or pause the current video if it was already playing.
Drag Right	Next Video	Plays next video in the playlist
Drag Left	Previous video	Plays the previous video in the screen

5.3 NFC interaction

NFC tags are located in the environment behind icons which advertise the command executed when a tag is touched with the mobile phone (see 3.3.4). The UI is embedded and distributed in the environment; users need to touch objects to trigger commands. Tags are placed wherever they are accessible by the users; on walls, posters, books, and so on. An NFC tag contains all the information necessary to create REACHeS HTTP request. Actually, the tag contains all URL parameters to be sent in the HTTP request. Optionally, it might include the URL of REACHeS server, when multiple REACHeS instances are running simultaneously.

One remarkable aspect of this interaction is that the context information (e.g. the location of the NFC tag) is also part of the UI. Each icon pictogram is linked to a command. The physical location of the tag determines the command parameters. For instance, a playroom can contain several teddy bears augmented with "play" icons. When a child touches any of these icons with a mobile phone, a video is shown in a close by display. Each teddy bear activates a different video. Touching the icon means "play a video" but it does not give information on what video should be played. It depends on the context. From REACHeS perspective, icons with the same pictogram contain identical `event` parameter. However, the rest of the URL parameters depend on the context in which the tag is placed. In the case of the teddy bear example, each teddy bear tag contains a different `playlist` parameter and even a different `target_resource` parameter if each toy plays the content in a different screen.

Both the mobile phone and the external resources provide feedback to users after touching an NFC tag. However, non-intrusive feedback related to the interaction itself must be provided by the mobile device; it is the device closer to the user. The phone should provide feedback just after the interaction (touching a tag) even before the command is sent to REACHeS. REACHeS clients use a combination of haptic and audio feedback to indicate whether a tag was read successfully or some error occurred. The aim of the feedback is that: (1) a user receives a confirmation that the area he/she touched with the mobile phone is an active area. If there is no feedback at all after touching an icon, the user learns that the touched area is not part of the application UI. (2) A user knows if the input was transmitted successfully to the system and if he/she needs to repeat the interaction. The tag might not be read correctly because the user touched the tag with the wrong part of the mobile phone or perform the touching action too quickly.

Based on usability tests, we learned that the mobile phone screen should be only used under three circumstances. Firstly, to show error messages when there are no suitable external resources available. Secondly, to use the phone's UI as a secondary input to the service (e.g. to enter a login or authentication information). This usage should be avoided and NFC should be preferred. Thirdly, the phone's screen can be used when there are no external resources available to show the expected content. Visual feedback on a phone's display should be minimized. If it is used, graphical icons and animations are better than text. The amount of information to be displayed on the screen must be minimized. Furthermore, visual information must always be accompanied with haptic or audio feedback to draw the user's focus of attention to the device's screen. On the other hand, external resources show the main application UI requiring user's focus of attention. The feedback of the mobile device should be minimized.

NFC-based interaction method provides seamless interaction between services and everyday objects. This method is less intrusive than gesture recognition and easier to use than classical keypad or touch interaction. A user does not need to browse through menus and complex UIs, but just touches objects in the environment. Furthermore, icon pictograms advertise the command to be executed when a mobile terminal touches them. The context (e.g. position of the tag) defines the parameters associated with the command. There is no need to recall any gesture or insert a set of configuration parameters as they are already stored in the NFC tag.

5.3.1 Starting the service

Icons distributed in the environment are used as hyperlinks to access services. When an icon is touched with a mobile phone REACHeS creates a new instance of the corresponding service, allocate necessary resources, set them up and finally start the client which transform the mobile device into a remote controller for the service. The tags contain the service that must be activated as well as its initial arguments. Multiple tags can coexist in an environment, each starting the same service with different parameters. The parameters might be general REACHeS arguments or specific for a service. For example, the tag information might define the resource allocation system (automatic, semiautomatic or manual) used to allocate resources, the preferred display or the playlists to be played.

As it is previously described in section 3.3.4 the design of the icons as well as their location in the environment are important factors to be considered when designing the application. In contrast to classical interfaces, the UI for services are distributed in the

environment. The icons that launch services must announce to the user (1) the service activated by this icon and (2) the particular arguments sent to the service when the tag is touched. Additionally, the position of the tag in the environment advertises information about the particular arguments stored in the tag. The same icon might activate the same service but with different initial conditions.

The great advantage of using this interaction mode to start a service is that users do not need to browse application menus to set up the initial arguments. For instance, in the Multimedia Player application, in a classical setup the user first starts the application and then selects from a list or menu the files to be played. Using this interaction mode the tag contains all the information.

5.3.2 Controlling the service

In this case all the commands that a service expects from a user are embedded into NFC tags. From the user perspective touching an icon is similar to press a button in a remote control for a TV or some other home appliance. Commands are sent to services after touching icons in the environment. We have named this interaction mode as Touch & Control.

Icons can be placed directly into objects or grouped into some artifacts that I generally name as “control boards”. The aim of a control board is to group all icons that are used in a specific task or application. A control board presents the available commands as control icons. The control board can have multiple realizations. Figure 36 shows some control boards for the Multimedia Player application. The appearance of a control board is really important. Users must perceive all the icons to be part of the same UI. For example, all the icons on Figure 36 share the same style and color.

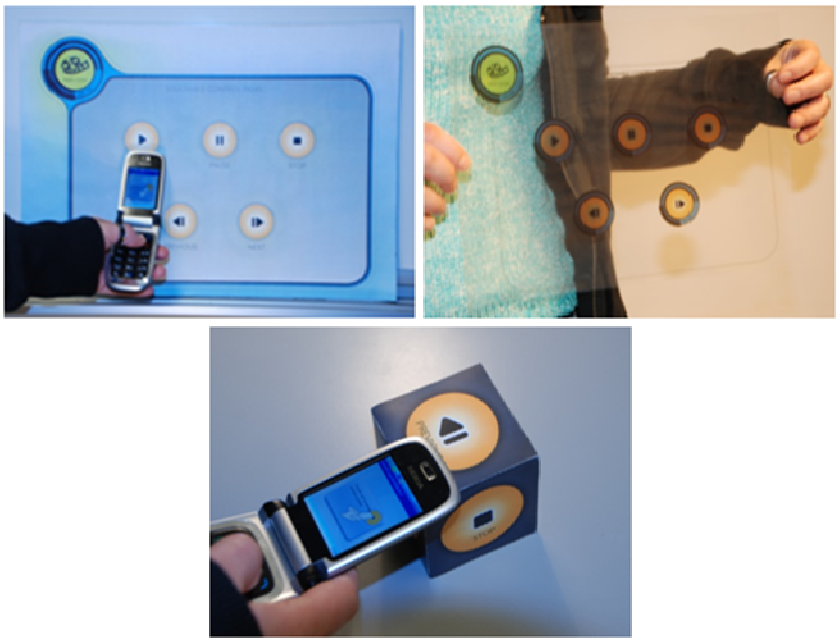


Figure 36. Control boards for Multimedia Player application. Each of them contains six icons: start/close, play, pause, stop, next and previous. The Control board on the upper left is an A3 paper glued to a thick cardboard. The board in the upper right is the same implementation but printed in a transparent paper. It is designed to be placed on a shop window so clients can control wall displays placed inside the shop. The figure at the bottom presents an alternative representation of the control board in the shape of a cube.

Touch&Control interaction offers several advantages over a mobile phone GUI. Firstly, the UI to control a service is not created on a display but it is embedded into environment. This fact makes it easy and cheap to personalize or update the UI. The UI is not coded in any software component, but the commands are stored in NFC tags. To modify an existing command (e.g. change some parameters) the NFC tag is rewritten. To add a new command to the environment, it suffices to write a new tag and placed it in the user environment. Furthermore, changing the style of the UI or the UI theme is as simple as replacing old icons by new ones. Moreover, the UI is easily reconfigurable. Changing the position of the icons generate a completely new UI. Moreover, one application can control multiple services. There is no need of an ad-hoc client for each service. Service-specific user interfaces are embedded in the environment. Secondly, Touch&Control allows a user to focus only in one screen. The phone's screen is just used to provide additional feedback. The user pays attention to the mobile phone screen occasionally. Moreover, the phone's keypad or touch screen is not used. Figure 37 provides some basic feedback shown on phone's display for Multimedia Player application. Thirdly, Touch&Control permits the use of big icons, improving the usability for visually impaired people. Icons might even contain Braille representations for blind people in addition to the visual symbol.

Figure 37. Visual feedback provided by the phone in the Multimedia Player application. The figure on the left instruct the user how to proceed after touching the start tag. Figure on the right shows feedback shown after user touches the play button.

The most relevant disadvantage of this method from the service design perspective is that the UI is static and cannot be modified depending on application state. Hence, not all type of applications can be implemented using exclusively this method.

5.3.3 Selecting devices

NFC tags attached to resources that are placed in the environment empower users to select manually which are the resources they desire to use. REACHeS uses this information during the resource allocation phase. If the allocation mode used by the particular service instance is "manual" then the mobile phone client provides the users with a list of resource types required to start the service. The user selects the resources from this list by touching the corresponding icon with the mobile phone. The tag under that icon contains the unique id of the resource inside REACHeS. The icon is a representation of the resource and it is placed in the resource itself (when it is accessible for the user) or in a control board or other representation for resources that are available but not directly accessible by the user (e.g. the speakers in a room might

be hanging from the ceiling, so user cannot access them physically). In this case the NFC tag does not contain the complete set of parameters to create the HTTP request to be sent to REACHeS, but the mobile client uses also other data to build the request. More information on this interaction mode can be found from [82].

5.3.4 Transferring content to resources

NFC tags placed on local resources are also a metaphor of a file container. A user can drop multimedia content on resources by touching the associated tag. A user perceives this interaction as if he is physically moving files from the mobile phone to the resource. I propose the following use case: user collects some material from the environment (e.g. taking some videos of an event). Using a mobile client she uploads this material to REACHeS. The user selects the files that she would like to watch in a big display and touches the icon attached to the display. The multimedia player application reproduces the desired videos. In this case, the content of the tag does not contain all the HTTP request parameters required for the service, but the mobile client adds the missing parameters (e.g. the playlist in the case of the example provided above) before sending the request to REACHeS.

5.4 Implemented services

5.4.1 Product Browser

This application permits users to browse advertisements on a wall display. Each advertisement shows a picture or video of a product, place or event, some text (maximum one line) and a URL to access more information about the product. A user browses different advertisements and opens the webpage to get more information. All the information is shown in the wall display. The phone's client uses the classical touch screen interaction to control the service (Figure 38). The UI is composed of two buttons to move to the next and previous advertisement and one button to access the webpage associated to the product. When this option is selected the UI in the client changes allowing the user to control the webpage vertical scroll. The user can go back to the default view by pressing the back button.

Figure 38. Product Browser UI. On the left two screens of the service client. On the right a view of the wall display.

One possible use case for this application is a travel agency advertising multiple holiday packages. The wall display is placed on a shop window or on the wall of the travel agency premises. The icon to start the application might be located in the shop window glass or in a poster close to the wall display. The tag behind the icon stores the id of such display and a playlist containing all the products to be shown as well as the multimedia content associated to it. The playlist uses the RSS 2.0 format.

5.4.2 Multimedia player for school environment.

A set of three different applications, namely multimedia player, REACHeS browser and picture carrousel were set up in a local high school. In this setup, students create their own multimedia content in form of video or picture set. Teachers create school news and event information in a website. Both teachers and students updated the content weekly. A projector is installed in the school main hall to project the material in a big screen of 55 inches. Students access the content by starting any of the three applications. The Multimedia Player plays the videos uploaded by the students, the Picture carrousel presents the picture set and finally the REACHeS browser shows the website uploaded by the teachers. Each application has its own mobile phone client. The multimedia player's client has already been described previously, the picture carrousel has two buttons: *next* and *previous* to browse the pictures and finally the REACHeS browser has arrows to control the browser scroll bar. All clients use the traditional interaction mode to control the content in the screen. A poster next to the screen is augmented with NFC tags which provide access to the services. The poster contains 5 tags. Two of them contain a playlist for the multimedia player, two of them a playlist for the picture carrousel while there is one tag to access the news webpage of the school. The tags were protected and their content could not be rewritten. The playlist was modified when new content was added to any of the applications.

The application was used during one month by 111 students. REACHeS services were running in our own server. The display was not only dedicated to REACHeS so the janitor had to unload and load the screen into REACHeS when it was needed. We created a very simple REACHeS display control webpage (Figure 39) to perform this task. Furthermore, we also deployed a simple content management application so teachers and students could upload new content to the applications as well as update the playlists. This feature was integrated into REACHeS and can now be used by any other service.

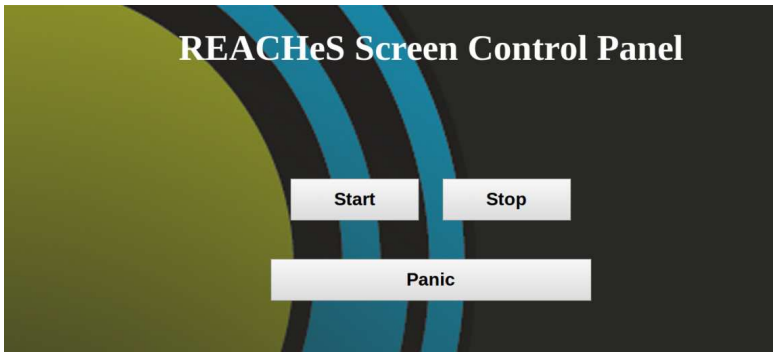


Figure 39. REACHeS control panel for loading and unloading displays in REACHeS. Panic button sends an email to application developer who will contact the person in charge.

5.4.3 Playlist manager and selector

This service was a proof of concept. The goal was to test (1) how REACHeS could implement a service that calls other REACHeS services and (2) how more complex NFC interaction works. This service improves the content management service implemented for the previous application. Users can upload and delete multimedia content and create an update playlists using a REACHeS customized web application (Figure 40 bottom). Both the playlist and the associated content can be accessed by any other REACHeS service. When a service requires a user to select a playlist manually, it launches the playlist selector service. User browses folders until he/she finds the desired playlist. When the user selects a playlist the file selector service forwards the chosen list to the caller service and exits. The caller service continues its process. The UI is very similar to traditional file system browser in modern operative systems (Figure 40).

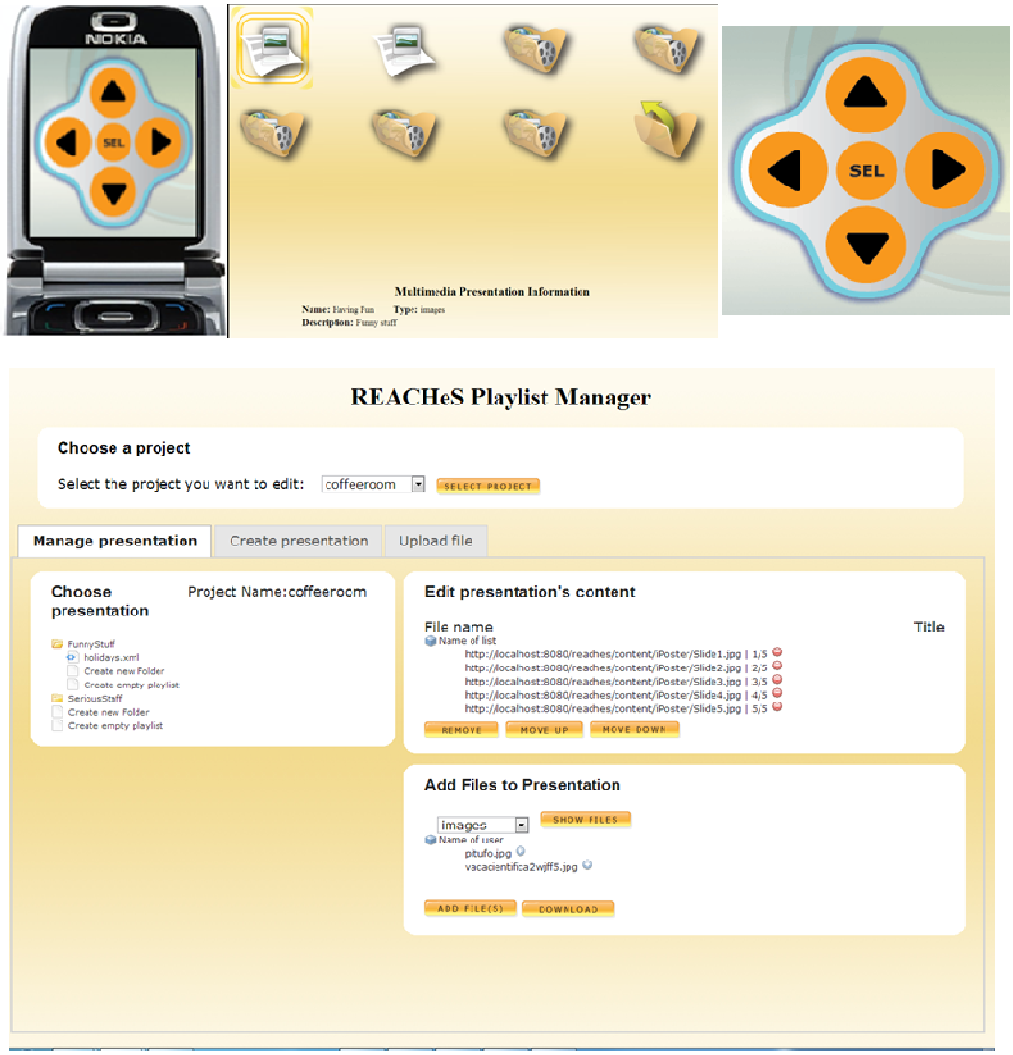


Figure 40. Playlist manager UI. On the upper left the UI for the mobile client using traditional interaction mode. When any physical key is pressed the corresponding icon is highlighted on the screen. The center shows a screenshot of the main UI screen. On the right a NFC augmented control board to control the service. At the bottom a view of the interface to create new playlists.

Playlists are stored in folders that at the same time might contain other folders. Icons embodying both playlists and folders are shown in the screen in a grid view. The

last element of the grid (bottom right) is the “back” icon, used to ascend in the file system hierarchy. One element is always in focus. The UI is shown in the main service display (not in the mobile phone screen) but it is controlled using the mobile phone. We tested two interaction modes for the device client: classical keypad and touchscreen and Touch&Control. In the first case, the user changes the focused element by pressing any of the arrows (up, down, left, right) of the keypad. By pressing the main action button users selects a playlist or accesses the folder content. A similar approach is used for Touch&Control. The UI is placed on a control board. Each tag stores a different REACHeS event (up, down, left, right and action). The control board can be placed for example in a poster advertising multiple services that needs to use the File Selector application. Figure 40 shows the UIs for the client.

5.4.4 Multimedia Center Control board

A Media Center is an application which permits listening to music and watching photographs and TV broadcasts on a TV or a computer. TAUCHI research group in Tampere have developed a Media Center application [87] focused on TV broadcast functionality. Their implementation allows browsing an EPG (Electronic Program Guide) in a Digital TV, searching for specific programs using multiple search criteria and scheduling the recording of one specific TV program. The UI is a grid where columns show TV channels, rows time slots and cells different TV programs. The cells are colored in several colors and have different icons depending on the type of program (news, TV, sports, etc.). TAUCHI group’s mobile client for this application used a combination of speech and gestures interaction. In addition, it is possible to navigate in the EPG with mobile phone’s arrows buttons. We integrated the Multimedia Center into REACHeS and use the Touch&Control approach to control the EPG. We built a control board with 33 different icons in an A3-size-carboard (Figure 41). This application proposes the following challenges: (1) Use of a big set of commands (and hence icons), (2) use of combined commands: those which are sent to REACHeS after a set of NFC tags have been touched and (3) integration with an external application developed by others.

The control board (Figure 41) has three different sections. Icons on the green area are general commands. They are sent to REACHeS instantly when the phone reads the tag content. They permit to change among different screens (EPG, TV and recorded videos), to zoom the EPG in and out, and to browse the TV programs of different days. Icons on the center (red, yellow and blue) filter the results of the EPG, highlighting only the programs that meet the filtering criteria. Each color represents a filtering criterion: yellow for TV channels, red for times of the day while blue for program types. A user composes the filter criteria by touching one icon in each area. For instance, show all the sports programs broadcasted during the night in MTV3. The filter command is not sent to REACHeS until the user presses the Send button on the mobile phone keypad and touch screen or touches the OK icon in the control board. Icons on the white area are higher-level commands that are replicated in the client UI. They send the filter command or clear the last icons touched by the user. The mobile phone display (Figure 41 right) shows the icons touched in a stack-like UI until the filter command is sent. The arrows permit selecting a program in the EPG without using the Touch&Control interaction mode.

Figure 41. The control board for the media center application (left). Phone's client (right) with three parameters in the start (sports in the morning in channel JIM).

5.4.5 Ubiquitous news reader

The Ubiquitous news reader application is used to study the different resource allocation mechanisms available for REACHeS and reported in [82]. The application is composed by two services: ubiquitous news reader and ubiquitous news player. NFC tags announcing the ubiquitous news reader are attached to a newspaper. The icons are hyperlinks to multimedia content (audio, picture or videos) associated to specific sections of the newspaper. The tags contain the necessary parameters to start the ubiquitous news reader application in the closest display. Each tag contains a different playlist. Hence, when an icon is touched with a mobile phone, the closest REACHeS display shows the list of the multimedia articles stored in the tag (Figure 42). A multimedia article is a combination of audio files, video and images. Audio narration provides the most important information, while videos and images are supplementary material whose role is to enrich user experience. An audio narration provides information which is normally printed in the newspaper. The narration consists of two parts the short and long version. The short version narrates the overview of the article (article's lead), while the long one contains the whole information (the story). Using the mobile phone as a remote controller a user browses the list and selects the multimedia content that he/she would like to visualize later. After entering in another room, and touching the ubiquitous news player service tag (placed usually next to the room's door), the user can watch or listen to the content that was previously collected from the newspaper.

This application shows how two services can be combined to produce a single application.

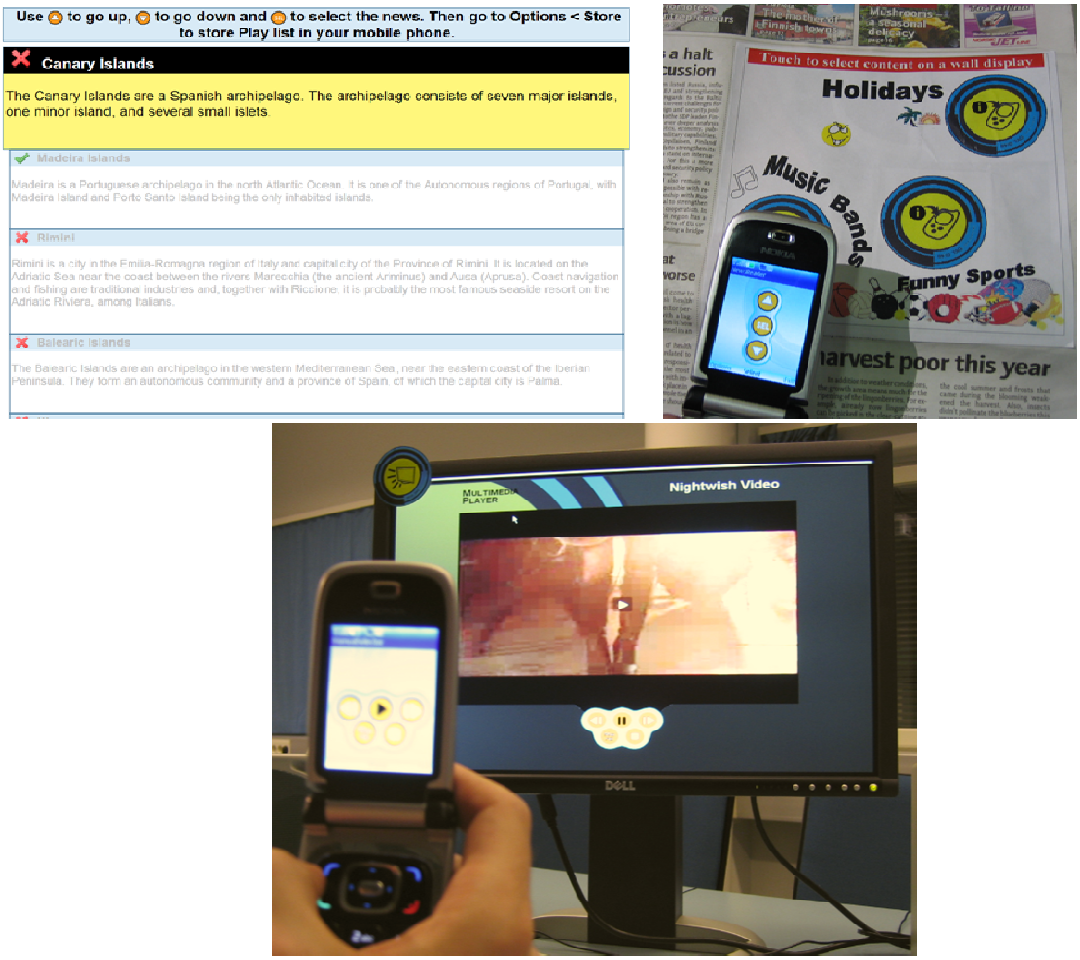


Figure 42 Ubiquitous news reader main UI (left). The user has selected an article related to Madeira Island. The client to control the ubiquitous news reader and the tags embedded in the newspaper are shown on the right. The UI in the main screen and mobile device client for the ubiquitous news player is seen at the bottom.

5.4.6 Interactive Poster

Interactive Poster [76] application is a set of services, activated by touching NFC tags attached to a conference poster. It provides an interactive experience to the visitors, extending the possibilities of a classical poster presentation. The services present multimedia content related to the poster either in the visitor's phone screen or in a nearby display. The services offered by Interactive Poster and activated by touching icons embedded in the poster are: listen poster abstract in the mobile device, send comments to the author, get multimedia content associated to the poster, pick poster's author's business card, suggest a meeting via SMS and reproduce multimedia content in a close by display. The only of these services that use REACHeS is the last one, in a similar way as the multimedia player for high school described in section 5.4.2.

The multimedia content to be reproduced is uploaded to REACHeS using the REACHeS playlist manager described in section 5.4.3. NFC tags (Figure 43) are attached to the poster. Each tag opens one of the following applications: Multimedia Player, REACHeS browser or picture carrousel. Icons are placed close to paragraphs or figures which are better supported by the multimedia content. Icons are printed in transparent plastic so they are not intrusive for the poster. NFC tags are glued to the back of the poster so they are not visible at all to the visitor.

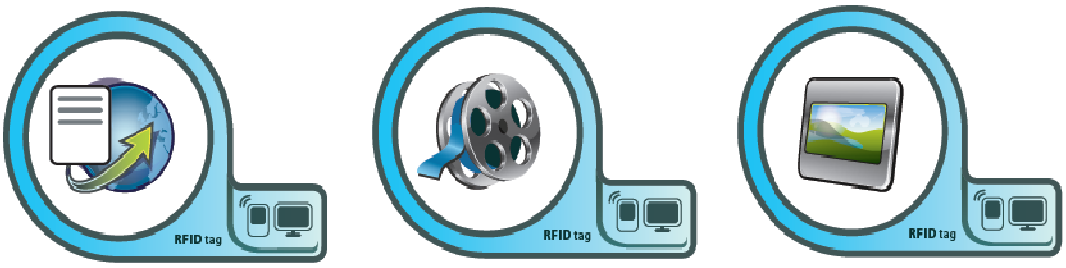


Figure 43. NFC icons placed in Interactive poster to access REACHeS browser (left), Multimedia Player (center) and picture carrousel (right)

6 USER AND PERFORMANCE STUDIES

We have lead several user studies in which prototypes were implemented using REACHeS. Several of the services described in section 5.4 have been tested by users in different environments. For example, the Multimedia Player for high school environment has been piloted in a high school for one month and has been used by 111 children. During Mindtrek conference (Tampere 2010) and Pervasive conference (Helsinki 2010) seven posters have been augmented with Interactive Poster application. Tens of visitors tested the application. Moreover the Multimedia Center control board and the News reader have been tested in a controlled lab environment (See sections 6.2.2 and 6.3). The usability studies that I describe in this section are focused on NFC interaction and technology acceptance and not in a concrete application. The user studies described in this section are not comprehensive enough but are preliminary studies which helps to evaluate the potential both for REACHeS and for the NFC based interaction. All the studies described in this section are already published in international conference papers, so I will remark the most important findings. For more information about the setup and results please have a look to the corresponding paper: [88], [89], [90], [82]

6.1 Touch & Control versus traditional keypad control

This test is presented in [88]. The goal of this test was to measure the usability of Touch&Control vs. traditional keypad control using the following usability metrics: reliability, easiness, perceived speed, intuitiveness and cognitive load. The usability test was performed in our premises. Before the test session we asked the test subjects to fill a demographic questionnaire. The test session took around 25 minutes per subject. In the beginning of the session the researcher gave a short introduction to the technology, the application and possible usage scenarios. Then the subject had to use the Multimedia Player application for 10 minutes using both interaction methods Touch & Control and traditional keypad control. REACHeS screen is loaded in a wall display. The laptop controlling the display was hidden to the test subject. The multimedia player was loaded with several playlists containing curious “Finnish cultural events”. Usage instructions for each one of the interaction methods were presented in an A4 page. The A4 paper sheet contained also the NFC tag to start the application using interaction mode object to study. The subject could use the application freely, but the researcher checked that the subject understood how the application works. The last ten minutes were reserved to fill a questionnaire and short discussion. There were always two researchers observing. The sessions were also recorded for future study

Ten subjects with ages between 23 and 44 years tested the application. All of them use smartphones normally and 7 of them were somehow familiar with RFID technology. Figure 44 shows the test setup while Table 16 summarizes the evaluation of different usability metrics given by the test subjects.

The users liked more Touch&Control than the traditional keypad GUI, at least for the type of application presented in the test. It had a better scoring in all usability metrics measured outstanding in intuitiveness and easiness. The low score in speed was probably due to technical reasons: mobile phone used in the test used GPRS technology to connect with REACHeS.

Figure 44. On the left test setup. On the right a test subject using the keypad control.

From the observation and video analysis we noted that users do not pay much attention to the feedback given on the mobile phone's display, instead they stay focused on the wall display. So, it is very important that feedback is somehow replicated on the main screen. Haptic feedback was also assessed as a very positive and necessary feedback. We also noticed that the touch interaction is very intuitive. Test subjects started to use Touch&Control without reading the instructions (after testing the traditional method first). Some users thought that one of the main advantages of touchable control panels is that they could move, so the users suggest that the control panels should not be fixed to any infrastructure but they should be mobile. Users thought that Touch&Control suits better public display scenario.

Table 16. Comparison of phone GUI vs. Touch & Control. Grades values range from 0 to 10

	PHONE GUI	**TOUCH & CONTROL**
Reliability	8,1	8,6
Easiness	8,7	9,4
Speed	6,8	7,6
Intuitiveness	8,4	9,2
Cognitive load	8,2	8,8
Average UX	8,35	8,6

6.2 Comparison in multimodal user interfaces

6.2.1 Gesture recognition vs. traditional keypad control.

This test is presented in [89]. In this test we compared three interaction modes to control an external display: gesture recognition (performing air gestures using a custom made wireless sensor device containing 3D acceleration sensors), using traditional mobile phone keypad as a remote control for the screen and using the touch capabilities of the wall display. The last one was used as a control experiment. During the test the subjects utilized a variation of the REACHeS browser application (see 5.4.2). In this case, with the mobile client the user could scroll up and down, move to the next and previous link of a webpage, click on a link and navigate through the browser history. The three interaction modes were supported by REACHeS. The final goal was to analyze the same usability metrics as in the previous test. We also asked the test subjects to order the different interaction methods in order of preference. Usability test was performed in our premises. The test session took around 20 minutes per subject. First the subjects filled a demographic questionnaire. Then the researcher introduced the technology. The test was divided in two different stages. During each stage the subjects perform the same task using the three interaction modes, being always the touch screen (control mode) the first one. The goal was to avoid bias because user did not know how to solve the task. During the first stage, the blind stage, the user was not instructed about how to use the application. The goal of this stage is measuring the intuitiveness of the interaction: how naturally the users utilize the system, and how well the users are able to discover the associated mappings from gestures and mobile phone's UI to browser commands. In the second stage: clear stage, the users were revealed the correct gestures and functionalities for each browsing command. We gave them a sheet of paper with the instructions. The goal of this stage was measure the usability and efficiency of the system. In both stages, the task was to answer a question given a specific page of Wikipedia as starting point. In the end the subjects filled in a questionnaire comparing the different interaction methods. Two researchers observed quietly the test while other was conducting it.

The user group consisted of eleven subjects, six male and five female with ages ranging 22-35. All of them were researchers at the Department of Computer and Electrical Engineering. During the blind stage, the users could map in average 4.5 icons (out of 8) on the mobile client to browsing commands and just 2.2 gestures. In the first case, we presume that the reason because the test subjects did not discovered all the commands was the big delay (<1s) between pressing a button in the mobile client and executing the command in the browser. Users thought that the button did not work and pressed another one. In gesture control we were formerly aware of the difficulties of associating commands to gestures. The easiest commands to discover were next link (move the sensor quickly to the right) and previous link (move the sensor quickly to the left). However, one of the test subjects managed to discover all the commands. It suggests that when a small set of gestures is discovered the rest of gestures are easy to deduce. During the second phase we measured the time to perform the task. We got similar results with both methods (gesture and phone keypad). The mobile phone clients as remote control took in average 2 min 19 sec while the gesture recognition 2 min 24 sec.

From observations, we noticed that users looked up the instructions sheet only testing the gesture case. The mobile phone control did not need instructions. On the other hand users did not feel comfortable changing constantly the focus of attention

between the mobile screen (to select a command) and the main screen (to check the effect of the command) in the mobile phone control's case. In the questionnaires, the mobile phone as remote control scored better in reliability, speed and intuitiveness while the gesture control was preferred when talking about handiness and low cognitive load. The easiness was evaluated equally for both interaction methods. In general, majority of users would opt for the mobile phone control for this application.

6.2.2 Speech and gesture recognition vs. Touch & Control (NFC)

Tampere University of Technology used REACHeS to compare their multimodal solution based on speech and gesture recognition with our Touch & Control approach to control the multimedia center described in section 5.4.4. I did not participate during the tests but in developing the interfaces and REACHeS services. In this section I will summarize the results reported by Kallinen et al. [90]. During the tests the subjects utilized both interaction methods to solve a set of specific tasks and exercises using the Media Center application. Before starting solving the tasks, subjects had to fill background questionnaire and a pre-test expectation questionnaire. The result of this questionnaire was compared with a post-test questionnaire to identify if perceived user experience is below their initial expectations. Instructions to use the different interaction modes were provided in a video.

Twenty subjects (with ages between 19 and 59 years old) participated in the study. Fifteen of them had experience with smart phones and 16 used RFID weekly. Results from tests shows that users perceived better user experience than they expected in the case of Touch & Control. It did not happen in the case of speech and gesture. Touch based interaction got the best evaluation. It was exciting, acceptable by others and it had clear affordances since the possibilities are visible. It was also considered easy to learn and rather innovative. The worse aspect evaluated by the users is the slowness of the touch interaction. However, quantitative data reveals that time-on-task for Touch&Control was 72.7% of the time-on-task for the speech and gesture user interface; hence Touch&Control interaction allows faster task completion than the other interaction mode. For measuring learnability the tester measured the time to perform two tasks that follows the same procedure. In the case of gestures and speech the second task was executed 44.4% faster than the first one while in the case of Touch&Control it was 52.3% faster. Thus, Touch&Control performs also better in learnability.

All in all, Touch&Control was better evaluated by users and got better quantitative results than speech recognition. For more details, refer to [90].

6.3 Comparison of resource allocation processes

In this test we tested the user perception of different resource allocation processes using the News reader application described in 5.4.3. The results and more detailed explanation of the system architecture and test setup are described in [82]. Two meeting rooms were converted into Interactive Spaces before the experiment. We placed in the rooms a set of REACHeS displays and speakers (Figure 45). In our laboratory lobby we set up a big display and a newspaper augmented with NFC tags. Each test session extended for around 1 hour. During the first minutes the test subject filled in a background questionnaire. After that we introduced the technology, the newsreader application and the test setup. Then, we invited the user to collect multimedia content from the newspaper located in the lobby and then visualize it in

any of the two Interactive Spaces. Each user had to use two of the REACHeS resource allocation methods (manual, automatic or semiautomatic). The methods that each user had to use were chosen by the researchers before the test. After finishing the tasks, the users filled in a questionnaire and commented the experience with the observers. We used the questionnaires to compare the interaction methods while the interview focused on collecting feedback on the concept and on the system in general. Two researchers were conducting the session and taking notes. We also record the test session in video for later study.

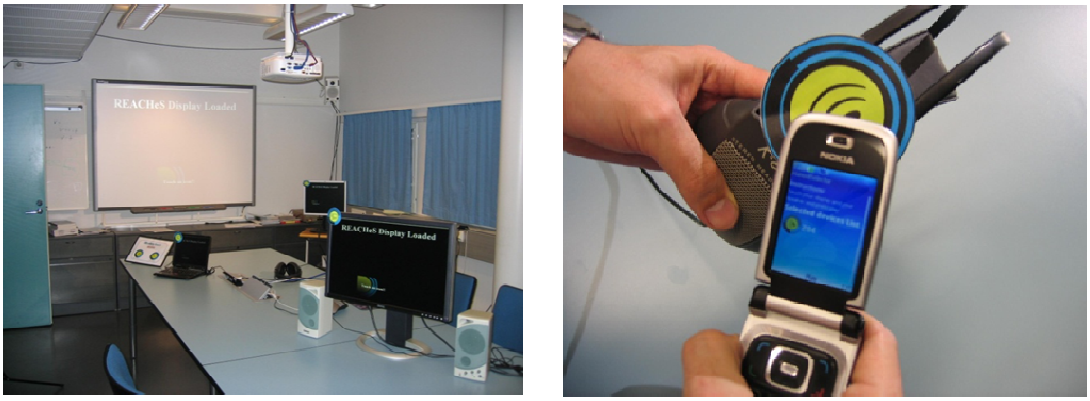


Figure 45 Test set up. On the left one of the room with 4 different displays and 4 different speakers (two of them are headphones). On the right a user selecting one of the speakers using the manual resource allocation method.

In total 30 subjects divided in 3 different groups (IT experts, savvy users and non-technical users). From the interviews we got the following key points:

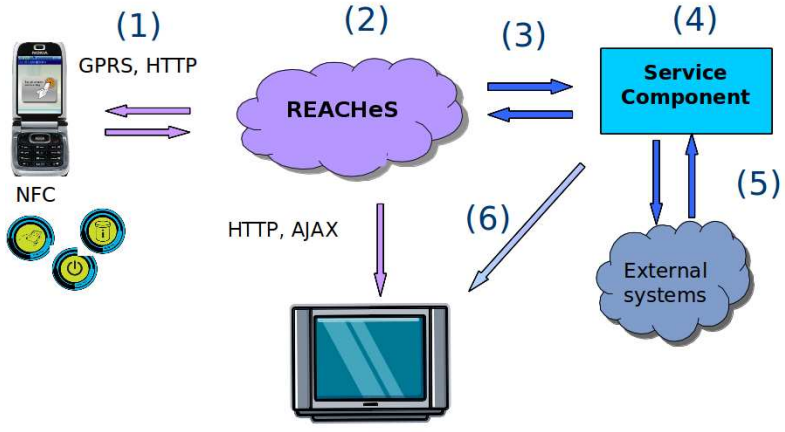
- **Manual method:** The three groups coincide that this method fits better in situations where user control is crucial and the reliability of the system is the most important factor. E.g. in a public presentation they would like to decide which display they would like to clone. This method was also preferred in private scenarios in which the user is familiar with the environment. In public spaces this method is preferred when privacy is required or users are involved in a social activity. They would not use this system to select resources that are not seen by the user (e.g. from which server you should receive the video streaming). Autonomous methods are better suited for that situation.
- **Semiautomatic method:** The most positive aspects of this method was that the user did not have to physically move to select a display (especially when the system selects a big display) or when they are in a public environment with a lot of people in the surroundings area. In the semiautomatic method the resource selection is performed in the screen of the user's mobile phone not visible to other people. IT experts said that they would rather use this method when they know beforehand why the systems choose those resources to use the service (they remark they want to be in control of the situation and they do not like unpredicted behavior). Other situation where users would rather use this method is when they are in a hurry.
- **Automatic method:** In general users were really reluctant to use this method since they did not have any possibility of modifying the decision made by the system. This is a really important finding when developing resource allocation systems for Interactive Spaces. The user should have always the possibility of modifying the

selection made by the system. They would use this automatic method if somehow the decision of the system is always predictable. However, this method had good acceptance in the non-technical subjects group. One positive aspect of this method remarked by all groups was easiness and quickness.

One aspect that attracted our attention is that non-technical users felt more confident with the automatic method while IT experts preferred semi-automatic or manual methods. Our initial hypothesis was the other way around. Other aspect that deserves attention is the fact that all user groups situated the manual method as the most intuitive one while the less intuitive was the automatic method. In contrast the automatic method was considered the easiest to use while the manual method was the most difficult one.

6.4 Performance test

We implemented a rudimentary performance test to identify possible time bottlenecks in REACHeS. From the user point of view it is really important that the time span since he presses a button in the phone keypad, touches an NFC tag or perform a gesture in the air until the command effect is shown/reproduced in the target resource is as short as possible. I used the picture carrousel application (described in 5.4.2) to measure different latencies. I used the fixed allocation method to allocate the target display. The mobile client was installed in a Nokia 6131 device. Nokia 6131 uses GPRS to connect to REACHeS. REACHeS is installed in our university network server, running on a Solaris machine. We tested two different version of picture carrousel one as REACHeS internal server and other as REACHeS external server. Display client was running on Firefox 2.0.0.4 installed on an Intel Pentium 4 3GHz with 2 GB of RAM and running Windows XP SP3. Display and REACHeS clock were synchronized using NTP protocol. In order to measure times in REACHeS I used some logging points in the applications, wrote timestamps in server and process the data using Matlab. I measured four different latencies (see Figure 46).



Round trip time
(1) + (2) + (3) + (4) + (5)
REACHeS execution time
(2) + (3) + (4) + (5)
Service execution time
(4) + (5)
Display update time
(6)

Figure 46. Latencies measured in REACHeS.

Round trip time is measured in the mobile client. It is the time since the HTTP is sent to REACHeS till the response is received. REACHeS execution time is the time measured since REACHeS receives the HTTP request until it sends back the HTTP response. The Service Component execution time elapsed since it receives the request

from REACHeS till it sends back the response. Finally, I also measured the time elapsed since the Service Component sends a command to the display till the command is executed. Moreover, when a wall display is used, the round trip time is not as important as the time elapsed from the moment the user sends an event till the event is executed on the wall display. I called this "effective time" and can be approximated from existing measures as follows:

$$Effective\ time\ = (Round\_trip\_time/\ 2)\ +\ Display\_update\_time$$

An application running in the mobile phone automatically starts the clients, sends 10 events to REACHeS and closes it. This process was repeated 50 times for internal services and 50 times for external services. The final results are shown in Table 17.

Table 17. REACHeS latencies. Start command distributions are not normal distributions.

	Average time (ms) Start command	**Average time (ms) Others command**
Round trip time (internal service)	3369 (σ = 2620)	1230 (σ = 705)
Round trip time (external service)	6600 (σ = 7697)	1141 (σ = 655)
REACHeS Execution time (internal service)	1265	10
REACHeS Execution time (external service)	1461	13
Service Execution time (internal service)	1095	2
Service Execution time (external service)	1076	5
Display update time (internal service)	1398	171
Display update time(external service)	1887	202
Effective time (internal service)	3082	786
Effective time(external service)	5187	772

The main bottleneck is in GPRS communication between REACHeS and the mobile phone. REACHeS execution time is 123 lower than the round trip time in the case of a standard commands for an internal service. Using faster connection technology such as Wi-Fi or 3G would reduce REACHeS latency considerably.

REACHeS execution time is around 120 times longer when processing the start command than processing another command. The start command generates a session and performs the resource allocation. Display update time is also around eight times bigger when processing a start command than when processing a standard command. The reason is that REACHeS display must load a new webpage in the browser when the start command is processed. Based on this data, to increase the efficiency of the system I should focus on improving the service initialization algorithms including the resource allocation methods. Writing and reading data from XML could be a reason of the bottleneck. Moving the database to a SQL or NoSQL database could partially solve this problem and increase REACHeS performance.

From the user experience point of view the most important measure is the Effective time. This is actually the time perceived by the user. Effective time is quite high in the case of the start command and acceptable in the case of other commands. However, the results of these tests shows that current REACHeS implementation is not ready yet for applications which needs fast interaction with the user such as action games. For all the applications proposed in this Thesis the resulting figures are quite acceptable to achieve a good user experience. The start command could be a problem though, due to the long waiting time. However, none of the users have complaint about that in any usability test. I assume that they are used to wait for an application to start in the mobile phone or in a desktop computer. I could use some tricks to reduce the perceived time especially in the case of the start command, though. For example, I could delay the moment in which the mobile phone start vibrating because it has read the tag and make the vibration longer (around 2s). The effect is that the user believes that the tag is still being read but in the background the HTTP request has already been sent.

7 DISCUSSION AND RELATED WORK

The goal of this thesis is to build a software platform (REACHeS) that enables quick implementation of services that can be integrated seamlessly into any environment (section 1.2). The aim of this work is to better understand Interactive Spaces. Near Field Communication is an emerging technology that we believe to be a strong candidate to be used as an enabler technology for Interactive Spaces. As such, REACHeS supports NFC natively in order to study when, where, and how NFC can be used to support Interactive Spaces. REACHeS platform glues together the different elements of an Interactive Space: users, services, resources (displays and speakers) and objects in the environment. Mobile phones are a familiar device for a large amount of people in developed countries. Furthermore, the mobility, computing power and network capabilities make mobile phones the most suitable tool to act as a mediator between the users, the environment and the REACHeS platform.

The concept of Interactive Spaces is not clearly defined as such in the literature. In this master thesis we sketch a possible definition of Interactive Spaces as the intersection among smart spaces, ubiquitous computing and physical user interfaces, emphasizing user control (instead of system autonomy). We also outline the main interaction modes and technologies for Interactive Spaces, focusing on tangible interaction since it permits embedding the UI in the environment and enables user control. However, the concept of Interactive Spaces is lacking a formal definition as well as a clear interaction model. The results of this thesis set the foundations to build a conceptual framework that helps to define better what Interactive Spaces are and what are the requirements and technologies to build those kinds of systems. A detailed analysis of the services defined, implemented and tested with real users and presented in this thesis will help in the future to complete this task. Tens of smart spaces realizations have been reported that might be considered Interactive Spaces using our definition. It is worth to mention one of the first smart spaces created for education: the Classroom 2000 project [91]. Moreover in the work reported by Johanson et al. [92] the staff of a company collaborate in a technology-rich space full of large displays and multimodal devices. However, these environments are not versatile but targeted to particular user scenarios.

REACHeS platform embeds services in the environment. The users control such services by interacting with objects in the environment using the mobile phone as a mediator: the mobile phone is a remote control for these services. The way of operating the phone depends on the interaction model selected. REACHeS initially was conceived as a simple platform for controlling local displays using featured mobile phones. An application is started by touching an NFC tag embedded in the environment. The tag data determines the display to be controlled and the service to be used. Functionality was added progressively to REACHeS in order to test different concepts related to Interactive Spaces. During the development we added support for new interaction modes for mobile clients (Touch&Control, gesture recognition), for multiple type of devices (client adaptation), for multiple simultaneous clients, for resource allocation (and its four allocation methods), for complex administration module, etc. All these features were not initially planned but were added after building and testing different services and applications. This indicates that the initial design was flexible enough for our research.

REACHeS bears some problems that make it difficult to be commercialized. The most important one is the latency: although its value is acceptable from the user

experience perspective, the latency of the whole system is expected to increase when multiple services, resources and clients are running in the same environment. The latency is already too big for real-time applications such as action games. However, latency was not a problem for any of the applications presented in this thesis. There are several ways of solving the latency problem: the first one is to use more advanced phones that support 3G, 4G or Wi-Fi communication. As shown in the performance tests REACHeS has a bottleneck in the air interface (between the phone and REACHeS). Another problem related to latency is the use of XML as repository. Using XML was useful in the initial design when supporting many devices, services and resources was not planned. XML enabled and off-the-shelf deployment to any server or computer. However, having the database in an XML file generates performance problems in speed, memory and CPU consumption. Future REACHeS versions should implement its repository in a SQL or a NOSQL database. A combination of them could be the best solution. SQL could be used for maintaining session information for clients, resources and services. NOSQL database could be used to store the description of resources and services. I would use SQlite for the first case and MongoDB for the second. SQlite is a very tiny SQL database with minimum setup while MongoDB is a document based database that uses JSON to store data.

REACHeS resource interface is another critical point from the efficiency and reliability point of view. The pushlet approach uses long polling HTTP requests. It is a valid solution but it consumes a lot of system resources. In a new version I would use a system that implements the publish/subscribe pattern using either Web Sockets or BOSH. The advantage of using BOSH is that it is valid for any kind of resource, and it does not need to run in a web browser. This implementation would permit the integration of other resources in the system, including actuators.

Another problem in current REACHeS implementation is scalability. Although in the tests the system behaves correctly with up to fifteen resources loaded, two services running and four clients registered simultaneously, I believe that the current implementation is not scalable when the number of users increases. An increase in the number of users means an increment in the number of services and resources running simultaneously. REACHeS centralized architecture does not help on that. A distributed architecture based on cloud services would improve REACHeS scalability. In a cloud based architecture we can clone the same REACHeS instance in multiple environments. Each environment runs its own REACHeS instance with its own database (that could be shared with other instances). Another alternative is to customize an existing smart space middleware and implement on it REACHeS functionality. One possible option is Smart-M3 [93].

On the other hand, the use of HTTP as the protocol to communicate between mobile phones, services, resources and REACHeS was a really good choice. Both HTTP addressability and uniform interface simplifies the addition of new features to REACHeS, especially when resources, services and REACHeS itself are running in different platforms. However, the overloaded POST method used in REACHeS is not the best option. A RESTful approach would have clarified the API making it easier to implement. In a RESTful approach the NFC tag would contain three different entities: an URL, an HTTP method and the body of the request. All in all, REACHeS has performed outstandingly for the purposes of the research carried out in this thesis and have operated always in a reliable fashion.

In the literature I could find several systems resembling REACHeS. The most similar one is Elope middleware [94]. Elope uses tagged physical objects and rooms to

integrate user generated content in the environment. Elope enables also the activation of services from tagged objects. The middleware links a device with certain content or with other device, setting up the parameters to start the communication. However, the platform does not support the control of the services that have been initialized. The mobile phone is not used to control a service but just to setup a communication between two devices. Resource allocation is performed manually by a user. Furthermore, it is not clear how to integrate new resources and services to the system. In Aura [95] the infrastructure adapts itself to the user context and needs. This system connects resources and environments among them. Aura performs resource allocation, but users are not involved in the decision process. A similar system, Gaia [96], is an operating system that facilitates the communication among resources to support the coordination of software entities sharing the same space. User interaction is not considered in the model either.

A key concept of REACHeS is the usage of a mobile phone to control resources. Others have reported systems with a similar idea: [97]–[100]. However, none of these publications present a general system that is capable of communicating services, resources and users. Broll et al. [101] presents a system to control the content of the phone display using web services to. In this work, as in REACHeS, web services are activated by interacting with objects in the environment. However, Broll's work does not include the control of external resources.

In the initial design, the phone browser is supported as one possible client. One big disadvantage of using the phone's browser is that the access to sensor information (NFC, accelerometer, etc.) is not allowed. Reading sensor data is only supported by native applications. Since a browser does not have access to sensor data it cannot be used as mobile phone client. This constraint could be overcome using a web runtime environment. Nowadays, web runtimes are available in all mobile platforms. The web runtime environment enables the integration of web applications in native applications. On the other hand, generating the UI in the server enhances the portability of the system. The mobile device does not need a different client application per service. The same application suffices to control all the services in the environment. However, this approach entails an unacceptable complexity for current REACHeS implementation. Hence, all REACHeS services were implemented using a native client. The same client could be used for several services, though. The whole UI is generated by the native application based on the service's responses. The big disadvantage of this approach is that a user needs to install a different client for each service or set of services that he/she is using. This is an interesting topic for future work.

REACHeS clients support multiple interaction methods by means of plugins (e.g. the gesture recognizer). However, this master thesis focuses on how to build UIs for Interactive Spaces using NFC technology. REACHeS initial design and the majority of interaction modes and applications presented in this thesis are based on NFC. Near Field Communication is a technology enabler to build physical user interfaces for Interactive Spaces. The interfaces are physical in the sense that mobile devices are used as physical objects to mediate between the user and the environment rather than as traditional input and output devices. We use also tangible user interfaces since the user interacts with services by manipulating objects by touching them with the mobile phone. The concept of embodied digital content in physical objects was presented by Ishii and Ullmer in their well-known publication Tangible bits [10]. The concept of digital embodiment implies the creation of "bridges" between the digital and the

physical content. In the seminal work of Want et al. [73], those bridges are built with RFID tags. REACHeS is an evolution of this concept. First of all, it uses a mobile phone as a mediator with the user and environment, and not a customized reader as Want uses in his research. From the user perspective, this is a big advantage since a user does not need to carry multiple devices. Besides, the sensors and actuators of mobile phones such as accelerometer, screen and speakers enable multimodal interaction. Another difference with Want et al. is that in our work user interaction is emphasized while Want's work focuses on technical aspects. Furthermore, in REACHeS, the same tag can be used by multiple services.

In REACHeS, tags are not only used to start services or link services to resources, but also to command them. A more thoroughly comparison of different interaction methods for Interactive Spaces (including NFC) has been carried out by Ailisto et al. [71] and Rukzio et al. [72]. Both compare three different interaction modes: touching, pointing and scanning. In REACHeS touching is emphasized over other interaction methods. Although, in this thesis, we have done a preliminary comparison with gesture interaction, a more comprehensive study is required. In general, usability tests show that NFC interaction is better accepted than gesture interaction for commanding resources. On the other hand, gestures offer better performance and versatility. As a result of the tests, we realized that the comparison of two interaction modes was not an easy task. The preference of one interaction mode over the other depends a lot on user personal preferences, context of usage and type of applications. A more interesting idea is to study how multiple interaction modes can be combined seamlessly to interact with services and resources (multimodal interaction). The same action can be achieved using different interaction modes. A user decides the mode to use based on his/her personal preferences. Studying multimodal interaction with NFC is more interesting than studying the interaction methods separately and then compare them. User interaction is a critical aspect of Interactive Spaces since those environments emphasize that the user is in control of the system at any time.

On a different matter, the existing literature proposes how to use NFC tags to start services or to connect services with resources [65], [71], [73], [102]. In our work we also present two other affordances that we could not find in the existing literature: tags to command services (Touch&Control) and tags as repositories of multimedia content. The Touch&Control concept has been tested in different scenarios. The user tests confirm a positive perception of this interaction mode. The tests have also demonstrated that we can build control boards with a large number of icons (33 were used in the media center's control board). These kinds of interfaces are much cheaper than classical displays or touch screens. A control board does not require technical skills to generate the UI since it is drawn on a paper and not coded in a device. Furthermore, NFC-based user interfaces are easily customizable and the UI can be distributed among objects in the environment. In addition, we believe that NFC has a huge potential among people with disabilities. Icons could contain Braille signs so blind users can interact with the control boards. Moreover, the usage of multiple tags to interact with services has been barely studied. We are only aware of Broll's work [103] in this field. He has studied the design of mobile and physical UIs for multi-tag interaction. Their results are hand in hand with ours. Multiple tags facilitate the selection of items or the accomplishment of complex tasks. It is simpler to use multiple tags for complex interactions than using the classical mobile phone keypad and screen. Users can focus on the environment and not on the menus or buttons in the phone.

The phone screen and speakers must be used to provide complementary feedback and contextual help to a user (e.g. indicating what actions he/she can do at a certain moment). However, interaction designers should take into account that users' attention is always focused on the environment. In the user tests we noticed that when an external display is used, the participants focused their attention mainly on that display. They did not pay attention to the mobile phone screen. Thus, mobile phones should be used as a secondary feedback source and the most relevant information should be provided in external displays. Besides, the visual feedback shown in the phone should be replicated in the external display. An exceptional case occurs when a user has to touch multiple tags sequentially to send a single command (see the Multimedia Center Control Board in section 5.4.4). In this case, the system informs to the user about the state of the interaction (e.g. showing the tags that he/she has previously touched) by using the mobile phone screen. In this case, the information provided in the mobile phone display is very important. We recommend avoiding this kind of situations and implementing services as stateless as possible.

In the tests, haptic feedback (vibration) has been revealed as a very effective feedback source. From vibration users know that they have sent a command to REACHeS without looking at the mobile phone screen. This feedback is crucial especially when there is a long latency between a user's phone and REACHeS.

NFC has some speed limitations when building UIs: processing the content of a NFC tag is not as fast as pressing a button or tapping a touch screen. So, NFC cannot be used to build interfaces in which the users must send multiple commands in a short period of time. I estimate that a minimum time between commands is around 1 s. Faster interaction does not perform well using control boards.

Services and commands are announced to the user by means of icons. An NFC tag is attached to the icon. The icon and the tag form together a two-sided interface between the physical and digital world. My colleagues have studied how to advertise the command associated to an icon and how the icons must be presented to the users so they perceive them as part of an application's UI. REACHeS has contributed in this study by enabling fast implementation of prototypes which led to a definition of a set design principles and guidelines [75]. There is not too much related work on this topic in the literature. Just a few publications refers to iconic representations [102], [104], [105] for icons.

Moreover, in this thesis I have studied just the reader/writer NFC operation mode. I consider that the P2P has a lot of potential, especially when the interaction among people in the environment is considered more important than the interaction with the environment itself. P2P mode has not been tested in any of the presented applications nor is supported by REACHeS clients. This is because we have focused our research in the interaction between users and environment and not in the interaction among humans. If we add users to the Interactive Spaces' interaction model NFC P2P interaction mode should be really considered.

All in all, the applications presented in this thesis have shown that NFC is a suitable technology to build Interactive Spaces. REACHeS have met all expectations, and it is a versatile system which facilitates the fast creation of services for Interactive Spaces.

8 CONCLUSION AND FUTURE WORK

The main contribution of this Master's Thesis is the design and implementation of REACHeS platform which permits controlling resources and services by using physical user interfaces. Although REACHeS clients are prepared to support multiple interaction methods, this research have focused on NFC interaction. The second contribution is new knowledge on how to use NFC technology to build UIs to interact with resources and applications. NFC is used to start and configure services, control services and select and interact with resources. The third contribution is a new definition of Interactive Spaces, a study of possible interaction modes and a survey of the technologies that can be used to build this kind of systems. Finally, we have built and tested with users several prototypes that have helped to understand better how to build user interfaces for Interactive Spaces using NFC technology.

There are several tracks to continue the work started in this thesis. One of them is to redefine the concept of Interactive Spaces, emphasizing two aspects: (1) users interact with the environment and not with a single device and (2) users are always in control of the interaction; the system does not initiate any task without a user's explicit intervention. Furthermore, another interesting research topic is to study how to integrate multimodal interaction with NFC. At this stage, NFC is studied as an interaction method isolated from others. But we claim that NFC has big potential to be integrated with other interaction modes. Moreover, the P2P mode has not been studied in this thesis and there is not much research in the literature either. We consider that the P2P mode has a high potential to be used as interaction method among people. Finally, REACHeS should be redesigned and implemented to solve the latency and scalability problems described in the section 7. Current REACHeS implementation and the knowledge obtained during the design and testing process create a solid foundation for these tasks.

9 BIBLIOGRAPHY

[1] Mark Weiser, 'The Computer for the 21st Century', *Scientific American*, vol. 265, pp. 94–104, 1991.

[2] Alessandro Valli, 'Natural Interaction White Paper'. 2007; http://www.naturalinteraction.org/images/whitepaper.pdf

[3] A. Greenfield, *Everyware : the dawning age of ubiquitous computing*. Berkeley CA: New Riders, 2006.

[4] A. Schmidt, 'Interactive Context-Aware Systems. Interacting with Ambient Intelligence', in *Ambient Intelligence*, G. Riva, F. Vatalaro, F. Davide, and M. Alcañiz, Eds. IOS press, 2005, pp. 159–178.

[5] A. Dey, 'Introduction to Context-Aware Computing', in *Ubiquitous Computing Fundamentals*, 1st ed., J. Krumm, Ed. CRC Press. Taylor & Francis Group, 2010, pp. 322 – 352.

[6] IST Advisory Group (ISTAG), 'Ambient Intelligence: from vision to reality'. 2003.

[7] S. Poslad, *Ubiquitous computing: smart devices, environments and interactions*. Chichester, U.K: Wiley, 2009.

[8] A. K. Dey and G. D. Abowd, 'Towards a better understanding of context and context-awareness', in *Computer Human Intraction 2000 Workshop on the What, Who, Where, When, Why and How of Context-Awareness*, 2000.

[9] T. Plötz, C. Kleine-Cosack, and G. Fink, 'Towards Human Centered Ambient Intelligence', in *Ambient Intelligence*, vol. 5355, E. Aarts, J. Crowley, B. de Ruyter, H. Gerhäuser, A. Pflaum, J. Schmidt, and R. Wichert, Eds. Springer Berlin / Heidelberg, 2008, pp. 26–43.

[10] H. Ishii and B. Ullmer, 'Tangible bits: towards seamless interfaces between people, bits and atoms', in *Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 1997, pp. 234–241.

[11] D. Norman, *The design of everyday things*, 1st Basic paperback. New York: Basic Books, 2002.

[12] K. P. Fishkin, A. Gujar, B. L. Harrison, T. P. Moran, and R. Want, 'Embodied user interfaces for *really* direct manipulation', *Commun. ACM*, vol. 43, no. 9, pp. 74–80, 2000.

[13] ITU-T, 'The Internet of Things. Executive Summary', 27441, 2005; http://www.itu.int/dms_pub/itu-s/opb/pol/S-POL-IR.IT-2005-SUM-PDF-E.pdf

[14] P. Stuckmann and R. Zimmermann, 'European research on future Internet design', *Wireless Communications, IEEE DOI - 10.1109/MWC.2009.5300298*, vol. 16, no. 5, pp. 14–22, 2009.

[15] K. Finkenzeller, *RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, 2003.

[16] S. Preradovic and N. C. Karmakar, 'RFID Transponders - A Review', in *International Conference on Electrical and Computer Engineering, ICECE '06*, 2006, pp. 96–99.

[17] T. Schaberreiter, C. Wieser, I. Sánchez, J. Riekki, and J. Röning, 'An Enumeration of RFID Related Threats', in *Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM '8)*, 2008, pp. 381–389.

[18] K. Mayes and K. Markantonakis, *Smart Cards, Tokens, Security and Applications*, 1st ed. Boston, MA: Springer Science Business Media, LLC, 2008.
[19] Z. Chen, *Java card technology for smart cards*. Boston [u.a.]: Addison-Wesley, 2000.
[20] W. Rankl, *Smart card handbook*, 3rd ed. Chichester West Sussex England ;Hoboken NJ USA: J. Wiley, 2003.
[21] Laran RFID, 'A basic introduction to RFID technology and its use in the supply chain. Whitepaper.', Apr. 2005; http://www.idii.com/wp/LaranRFID.pdf
[22] W. Matt, R. van Kranenbury, and G. Backhouse, 'RFID: Frequency, standards, adoption and innovation', *JISC Technology and Standards Watch,*, May 2006; http://www.jisc.ac.uk/media/documents/techwatch/tsw0602.pdf
[23] H. Witschnig, C. Patauner, A. Maier, E. Leitgeb, and D. Rinner, 'High speed RFID lab-scaled prototype at the frequency of 13.56 MHz', *e & i Elektrotechnik und Informationstechnik*, vol. 124, pp. 376–383, Nov. 2007.
[24] T. Salminen, S. Hosio, and J. Riekki, 'Enhancing Bluetooth Connectivity with RFID', in *Pervasive Computing and Communications*, 2006, pp. 36–41.
[25] The Globe and Mail, 'Near future of near field', 2007. [Online]. Available: http://www.theglobeandmail.com/news/technology/article781161.ece. [Accessed: 14-Sep-2011].
[26] NFC Forum, 'Near Field Communication and the NFC Forum: The Keys to Truly Interoperable Communications'. 2007. [Online]. Avaialble: http://www.nfc-forum.org/resources/white_papers/nfc_forum_marketing_white_paper.pdf [Accessed 10.10.2013]
[27] A. Mitrokotsa, M. R. Rieback, and A. S. Tanenbaum, 'Classifying RFID attacks and defenses', *Information Systems Frontiers*, vol. 12, no. 5, pp. 491–505, Jul. 2009.
[28] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis, 'Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones', in *Radio Frequency Identification: Security and Privacy Issues*, vol. 6370, S. B. Ors Yalcin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–49.
[29] E. Haselsteiner and K. BeritfuB, 'Security in Near Field Communication (NFC)', in *Workshop on RFID Security*, Graz, Austria, 2006.
[30] C. Mulliner, 'Vulnerability Analysis and Attacks on NFC-enabled Mobile Phones', in *Proceedings of the International Conference on Availability Reliability and Security*, Fukuoka, Japan, 2009, pp. 695–700.
[31] M. Roland, J. Langer, and J. Scharinger, 'Security Vulnerabilities of the NDEF Signature Record Type', in *3rd International Workshop on Near Field Communication (NFC), 2011*, 2011, pp. 65–70.
[32] J. Häikiö, A. Wallin, M. Isomursu, H. Ailisto, T. Matinmikko, and T. Huomo, 'Touch-based user interface for elderly users', in *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, 2007, pp. 289–296.
[33] M. Ervasti, M. Isomursu, and M. Kinnula, 'Experiences from NFC Supported School Attendance Supervision for Children', in *In Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2009, pp. 22–30.
[34] Tuikka T and Isomursu M, 'Touch the future with a smart touch', *VTT Research Notes*, no. 2492, 2009.

[35] H. Ailisto, M. Isomursu, T. Tuikka, and J. Häikiö, ‘Experiences from interaction design for NFC applications’, *J. Ambient Intell. Smart Environ.*, vol. 1, no. 4, pp. 351–364, Dec. 2009.
[36] I. Sánchez, J. Riekki, J. Rousu, and S. Pirttikangas, ‘Touch & Share: RFID based ubiquitous file containers’, in *Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia*, Umeå, Sweden, 2008, pp. 57–63.
[37] E. Siira, T. Tuikka, and V. Tormanen, ‘Location-Based Mobile Wiki Using NFC Tag Infrastructure’, in *First International Workshop on Near Field Communication*, 2009, pp. 56 –60.
[38] S. Miranda and N. Pastorelly, ‘NFC Mobiquitous Information Service Prototyping at the University of Nice Sophia Antipolis and Multi-mode NFC Application Proposal’, in *3rd International Workshop on Near Field Communication (NFC)*, 2011, pp. 3 –8.
[39] I. Luque Ruiz, P. Castro Garrido, G. Matas Miraz, F. Borrego-Jaraba, and M. Á. Gómez-Nieto, ‘University of Things. Toward the Pervasive University.’, in *Near field communications handbook*, vol. 13, S. Ahson and M. Ilyas, Eds. Boca Raton: CRC Press, 2012.
[40] P. Menschner, A. Prinz, P. Koene, F. Köbler, M. Altmann, H. Krcmar, and J. Leimeister, ‘Reaching into patients’ homes – participatory designed AAL services’, *Electronic Markets*, vol. 21, no. 1, pp. 63–76, 2011.
[41] J. Riekki, S. Sasin, and S. Pirttikangas, ‘Touchnrun: An RFID-based Distributed Board Game Motivating to Move’, in *Poster Proceedings Persuasive Technology*, 2008, pp. 34 – 40.
[42] G. Broll, R. Graebsch, P. Holleis, and M. Wagner, ‘Touch to play: mobile gaming with dynamic, NFC-based physical user interfaces’, in *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, New York, NY, USA, 2010, pp. 459–462.
[43] F. Resatsch, S. Karpischek, U. Sandner, and S. Hamacher, ‘Mobile sales assistant: NFC for retailers’, in *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, New York, NY, USA, 2007, pp. 313–316.
[44] A. J. Ingle and B. W. Gawali, ‘Status of Wireless Technologies Used For Designing Home Automation System - A review’, *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 7, pp. 142 –146, Jul. 2011.
[45] M. Weik, ‘The ENIAC Story’, 1961. [Online]. Available: http://ftp.arl.mil/~mike/comphist/eniac-story.html. [Accessed: 13-Nov-2013].
[46] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-computer interaction*. Harlow, England; New York: Pearson/Prentice-Hall, 2003.
[47] A. Quigley, ‘From GUI to UUI: Interfaces for Ubiquitous Computing’, in *Ubiquitous Computing Fundamentals*, 1st ed., J. Krumm, Ed. CRC Press. Taylor & Francis Group, 2010, pp. 237–283.
[48] J. Preece, Y. Rogers, and H. Sharp, *Interaction design : beyond human-computer interaction*. New York, NY: J. Wiley & Sons, 2002.
[49] P. Kortum, *HCI beyond the GUI design for haptic, speech, olfactory and other nontraditional interfaces*. Amsterdam; Boston: Elsevier/Morgan Kaufmann, 2008.

[50] M. Liljedahl, S. Lindberg, K. Delsing, M. Polojärvi, T. Saloranta, and I. Alakärppä, ‘Testing Two Tools for Multimodal Navigation’, *Advances in Human-Computer Interaction*, vol. 2012, pp. 1–10, 2012.
[51] J. Alty, ‘Multimedia-What is it and how do we exploit it’, *Proceedings of HCI*, pp. 31–44, Aug. 1991.
[52] M. Gullberg and K. Holmqvist, ‘What speakers do and what addressees look at: Visual attention to gestures in human interaction live and on video’, *Pragmatics & Cognition*, vol. 14, no. 1, pp. 53–82, Jan. 2006.
[53] V. I. Pavlovic, R. Sharma, and T. S. Huang, ‘Visual interpretation of hand gestures for human-computer interaction: A review’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 677–695, 1997.
[54] D. A. Norman, ‘Natural user interfaces are not natural’, *Interactions*, vol. 17, no. 3, p. 6, May 2010.
[55] D. A. Norman and J. Nielsen, ‘Gestural Interfaces: A Step Backward In Usability’, *Interactions*, vol. 17, no. 5, p. 46, Sep. 2010.
[56] L. M. Reeves, J.-C. Martin, M. McTear, T. Raman, K. M. Stanney, H. Su, Q. Y. Wang, J. Lai, J. A. Larson, S. Oviatt, T. S. Balaji, S. Buisine, P. Collings, P. Cohen, and B. Kraal, ‘Guidelines for multimodal user interface design’, *Communications of the ACM*, vol. 47, no. 1, p. 57, Jan. 2004.
[57] L. A. Jones and N. B. Sarter, ‘Tactile Displays: Guidance for Their Design and Application’, *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 50, no. 1, pp. 90–111, Feb. 2008.
[58] D. Spelmezan, M. Jacobs, A. Hilgers, and J. Borchers, ‘Tactile motion instructions for physical activities’, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 2243–2252.
[59] J. van der Linden, E. Schoonderwaldt, J. Bird, and R. Johnson, ‘MusicJacket. Combining Motion Capture and Vibrotactile Feedback to Teach Violin Bowing’, *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 1, pp. 104–113, Jan. 2011.
[60] K. Maclean and M. Enriquez, ‘Perceptual design of haptic icons’, in *In Proceedings of Eurohaptics*, 2003, pp. 351–363.
[61] U. Rashid and A. Quigley, ‘Ambient Displays in Academic Settings: Avoiding their Underutilization’, in *Proc. of 2nd Workshop on Ambient Information Systems. Colocated with Ubicomp*, 2008.
[62] D. W. F. Van Krevelen and R. Poelman, ‘A survey of augmented reality technologies, applications and limitations’, *International Journal of Virtual Reality*, vol. 9, no. 2, p. 1, 2010.
[63] C. Müller-Tomfelde and M. Fjeld, ‘Tabletops: Interactive Horizontal Displays for Ubiquitous Computing’, *Computer*, pp. 78–81, 2012.
[64] M. Kaltenbrunner, S. Jorda, G. Geiger, and M. Alonso, ‘The reactable*: A collaborative musical instrument’, in *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2006, pp. 406–411.
[65] K. Seewoonauth, E. Rukzio, R. Hardy, and P. Holleis, ‘Touch & connect and touch &select: interacting with a computer by touching it with a mobile phone’, in *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, New York, NY, USA, 2009, pp. 36:1–36:9.

[66] G. Broll, W. Reithmeier, P. Holleis, and M. Wagner, ‘Design and evaluation of techniques for mobile interaction with dynamic NFC-displays’, in *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, 2011, pp. 205–212.
[67] B. Ullmer and H. Ishii, ‘Emerging frameworks for tangible user interfaces’, in *Human-computer interaction in the new millennium*, J. Carroll, Ed. New York: Addison-Wesley, pp. 579–601.
[68] K. P. Fishkin, ‘A taxonomy for and analysis of tangible interfaces’, *Personal and Ubiquitous Computing*, vol. 8, no. 5, pp. 347–358, Sep. 2004.
[69] E. Hornecker and J. Buur, ‘Getting a grip on tangible interaction: a framework on physical space and social interaction’, in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, New York, NY, USA, 2006, pp. 437–446.
[70] O. Shaer and E. Hornecker, ‘Tangible User Interfaces: Past, Present and Future Directions’, *Foundations and Trends in Human-Computer Interaction*, vol. 3, no. 1–2, pp. 1 – 137, Feb. 2010.
[71] H. Ailisto, L. Pohjanheimo, P. Välkkynen, E. Strömmer, T. Tuomisto, and I. Korhonen, ‘Bridging the physical and virtual worlds by local connectivity-based physical selection’, *Personal and Ubiquitous Computing*, vol. 10, no. 6, pp. 333–344, 2006.
[72] E. Rukzio, K. Leichtenstern, V. Callaghan, P. Holleis, A. Schmidt, and J. Chin, ‘An Experimental Comparison of Physical Mobile Interaction Techniques: Touching, Pointing and Scanning’, in *Proceedings on the 8th Intl Conf. Ubiquitous Computing*, 2006, vol. 4206, pp. 87–104.
[73] R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison, ‘Bridging physical and virtual worlds with electronic tags’, in *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, Pittsburgh, Pennsylvania, United States, 1999, pp. 370–377.
[74] J. Riekki, I. Sánchez, and M. Pyykkönen, ‘Remote control for pervasive services’, *International Journal of Autonomous and Adaptive Communications Systems*, vol. 3, no. 1, pp. 39 – 58, 2010.
[75] M. Pyykkönen, J. Riekki, I. Alakärppä, I. Sanchez, M. Cortes, and S. Saukkonen, ‘Designing Tangible User Interfaces for NFC Phones’, *Advances in Human-Computer Interaction*, vol. 2012, pp. 1–12, 2012.
[76] I. Sanchez, M. Cortes, J. Riekki, and M. Pyykkönen, ‘NFC-Based Physical User Interfaces for Interactive Spaces’, in *Near field communications handbook*, S. Ahson and M. Ilyas, Eds. Auerbach Publications, 2011, pp. 175–230.
[77] T. Diekmann, A. Melski, and M. Schumann, ‘Data-on-Network vs. Data-on-Tag: Managing Data in Complex RFID Environments’, in *40th Annual Hawaii International Conference on System Sciences*, Hawaii, 2007, p. 224a.
[78] G. Broll, S. Keck, P. Holleis, and A. Butz, ‘Improving the accessibility of NFC/RFID-based mobile interaction through learnability and guidance’, in *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, New York, NY, USA, 2009, pp. 37:1–37:10.
[79] A. Hang, G. Broll, and A. Wiethoff, ‘Visual design of physical user interfaces for NFC-based mobile interaction’, in *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, New York, NY, USA, 2010, pp. 292–301.

[80] F. Köbler, P. Koene, F. Resatsch, J. Leimeister, and H. Krcmar, ‘NFC Application design guidelines’, in *Near field communications handbook*, S. Ahson and M. Ilyas, Eds. Auerbach Publications, 2011, pp. 91–115.
[81] L. Richardson and S. Ruby, *RESTful web services*. Farnham: O’Reilly, 2007.
[82] O. Davidyuk, I. Sanchez, and J. Riekki, ‘CADEAU: Supporting Autonomic and User-Controlled Application Composition in Ubiquitous Environments’, *Pervasive Computing and Communications Design and Deployment: Technologies, Trends, and Applications*, vol. 4, pp. 74–102, 2011.
[83] O. Davidyuk, I. Sánchez, J. I. Duran, and J. Riekki, ‘Autonomic composition of ubiquitous multimedia applications in reaches’, in *Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia*, 2008, pp. 105–108.
[84] O. Davidyuk, I. Selek, J. Imanol Duran, and J. R. Riekki, Jukka, ‘Algorithms for Composing Pervasive Applications’, *International Journal of Software Engineering and Its Applications*, vol. 2, no. 2, pp. 71–94, 2008.
[85] S. Bordet, ‘The CometD Reference Book’. 2011; http://docs.cometd.org/reference/
[86] M. Kauppila, S. Pirttikangas, X. Su, and J. Riekki, ‘Accelerometer based gestural control of browser applications’, in *Proceedings of the International Workshop on Real Field Identification*, 2007, pp. 2–17.
[87] M. Turunen, A. Kallinen, I. Sánchez, J. Riekki, J. Hella, T. Olsson, A. Melto, J.-P. Rajaniemi, J. Hakulinen, E. Mäkinen, P. Valkama, T. Miettinen, M. Pyykkönen, T. Saloranta, E. Gilman, and R. Raisamo, ‘Multimodal interaction with speech and physical touch interface in a media center application’, in *Proceedings of the International Conference on Advances in Computer Enterntainment Technology*, Athens, Greece, 2009, pp. 19–26.
[88] I. Sánchez, J. Riekki, and M. Pyykkönen, ‘Touch & Control: Interacting with Services by Touching RFID Tags’, in *2nd International Workshop on RFID Technology - Concepts, Applications, Challenges. IWRT 2008*, Barcelona, Spain, 2008, pp. 53–62.
[89] S. Pirttikangas, I. Sánchez, M. Kauppila, and J. Riekki, ‘Comparison of Touch, Mobile Phone, and Gesture Based Controlling of Browser Applications on a Large Screen’, in *Adjunct Proceedings Pervasive 2008*, Sydney, Australia, 2008, pp. 5–8.
[90] A. Kallinen, ‘Studies on User Experience of Touch-Based Interaction’, Tampere University of Technology, Tampere, 2010.
[91] G. D. Abowd, ‘Classroom 2000: An experiment with the instrumentation of a living educational environment’, *IBM Systems Journal*, vol. 38, no. 4, pp. 508–530, 1999.
[92] B. Johanson, A. Fox, and T. Winograd, ‘The Interactive Workspaces project: experiences with ubiquitous computing rooms’, *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 67–74, Apr. 2002.
[93] J. Honkola, H. Laine, R. Brown, and O. Tyrkko, ‘Smart-M3 information sharing platform’, in *IEEE Symposium on Computers and Communications*, 2010, pp. 1041–1046.
[94] T. Pering, R. Ballagas, and R. Want, ‘Spontaneous marriages of mobile devices and interactive spaces’, *Communications of the ACM*, vol. 48, no. 9, p. 53, Sep. 2005.

[95] D. Garlan, D. P. Siewiorek, A. Smailagic, and P. Steenkiste, ‘Project Aura: toward distraction-free pervasive computing’, *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 22–31, Apr. 2002.
[96] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, ‘Gaia: a middleware platform for active spaces’, *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 4, pp. 65–67, Oct. 2002.
[97] B. A. Myers, ‘Mobile Devices for Control’, in *Human Computer Interaction with Mobile Devices*, vol. 2411, F. Paternò, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–8.
[98] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan, ‘The Smart Phone: A Ubiquitous Input Device’, *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 70–77, Jan. 2006.
[99] J. Nichols and B. A. Myers, ‘Controlling Home and Office Appliances with Smart Phones’, *IEEE Pervasive Computing*, vol. 5, no. 3, pp. 60–67, Jul. 2006.
[100] M. Raghunath, N. Ravi, M.-C. Rosu, and C. Narayanaswami, ‘Inverted Browser: A Novel Approach towards Display Symbiosis’, pp. 71–76.
[101] G. Broll, S. Siorpaes, E. Rukzio, M. Paolucci, J. Hamard, M. Wagner, and A. Schmidt, ‘Supporting Mobile Service Usage through Physical Mobile Interaction’, 2007, pp. 262–271.
[102] J. Riekki, T. Salminen, and I. Alakarppa, ‘Requesting Pervasive Services by Touching RFID Tags’, *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 40–46, Jan. 2006.
[103] G. Broll and D. Hausen, ‘Mobile and physical user interfaces for NFC-based mobile interaction with multiple tags’, in *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, Lisbon, Portugal, 2010, pp. 133–142.
[104] M. Tungare, P. S. Pyla, P. Bafna, V. Glina, W. Zheng, X. Yu, U. Balli, and S. Harrison, ‘Embodied data objects: tangible interfaces to information appliances’, in *Proceedings of the 44th annual Southeast regional conference*, 2006, pp. 359 – 364.
[105] P. Välkkynen, T. Tuomisto, and I. Korhonen, ‘Suggestions for Visualising Physical Hyperlinks’, in *Workshop Proceedings of Pervasive Mobile Interaction Devices*, 2006.